On Enhancing the Robustness Of Timeline Summarization Test Collections

RICHARD MCCREADIE, University of Glasgow
SHAHZAD RAJPUT, National Institute of Standards and Technology
IAN SOBOROFF, National Institute of Standards and Technology
CRAIG MACDONALD, University of Glasgow
IADH OUNIS, University of Glasgow

Timeline generation systems are a class of algorithms that produce a sequence of time-ordered sentences or text snippets extracted in real-time from high-volume streams of digital documents (e.g. news articles), focusing on retaining relevant and informative content for a particular information need (e.g. topic or event). These systems have a range of uses, such as producing concise overviews of events for end-users (human or artificial agents). To advance the field of automatic timeline generation, robust and reproducible evaluation methodologies are needed. To this end, several evaluation metrics and labeling methodologies have recently been developed - focusing on information nugget or cluster-based ground truth representations, respectively. These methodologies rely on human assessors manually mapping timeline items (e.g. sentences) to an explicit representation of what information a 'good' summary should contain. However, while these evaluation methodologies produce reusable ground truth labels, prior works have reported cases where such evaluations fail to accurately estimate the performance of new timeline generation systems due to label incompleteness. In this paper, we first quantify the extent to which the timeline summarization test collections fail to generalize to new summarization systems, then we propose, evaluate and analyze new automatic solutions to this issue. In particular, using a depooling methodology over 19 systems and across three high-volume datasets, we quantify the degree of system ranking error caused by excluding those systems when labeling. We show that when considering lower-effectiveness systems, the test collections are robust (the likelihood of systems being miss-ranked is low). However, we show that the risk of systems being mis-ranked increases as the effectiveness of systems held-out from the pool increases. To reduce the risk of mis-ranking systems, we also propose a range of different automatic ground truth label expansion techniques. Our results show that the proposed expansion techniques can be effective at increasing the robustness of the TREC-TS test collections, as they are able to generate large numbers missing matches with high accuracy, markedly reducing the number of mis-rankings by up to 50%.

Author Note: This article has been accepted for publication in the Journal of Information Processing and Management: Special Issue on Narrative Extraction (Text2Story). This is an extension of the full paper 'Automatic Ground Truth Expansion for Timeline Evaluation' published in the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018. [https://dl.acm.org/citation.cfm?id=3210034]

Publisher DOI: [https://doi.org/10.1016/j.ipm.2019.02.006].

1 INTRODUCTION

With the increasing usage of social media platforms and online reporting channels, information is produced and disseminated online faster and in larger volumes than ever before. As a result, users expect to have easy access to up-to-date information about topics of interest, resulting in a large number of new real-time information-seeking scenarios. These scenarios require solutions that can identify relevant (topical), non-redundant (avoids repeated information), and timely (up-to-date) content from noisy high-volume text streams. A common class of solutions that require these characteristics are event timeline/real-time summary generation systems. Such systems take as input a topic of interest and a large volume of textual items (e.g. news articles or tweets), most of

Table 1. Example timeline summary extract with nuggets.

Timestamp	Timeline Updates	Information Units (Nuggets)
01/14/2012, 5:02pm	Carrying 3,206 passengers and 1,023 crew members, the Costa Concordia was on its usual route across the Mediterranean Sea and departed Civitavecchia - three hours before disaster struck.	Crew and Passenger count, Ship route, Time of departure
01/14/2012, 9:38pm	As the Costa Concordia keeps shifting on its rocky ledge, many have raised the prospect of a possible environmental disaster if the 2,300 tons of fuel on the half-submerged cruise ship leaks into the sea.	Fuel oil environmental hazard
01/15/2012, 5:17pm	The Costa Concordia death toll has risen by two - as all British passengers and crew were confirmed to have survived. Two French nationals and a Peruvian died after the cruiser ran aground near the island of Giglio off the Tuscan coast on Friday night.	People killed increased by 2. Location of event

which are non-relevant and/or redundant, and select a novel and relevant subset of those items to be emitted over time into a timeline or an updating summary [17, 32]. An example extract from the output of such systems is shown in Table 1. Such systems are useful for producing concise overviews of events in real-time, such as 'key points' content shown on news websites like the BBC.com for live stories, or to create a curated stream of updates for a user during events [45]. These systems can also be used over retrospective datasets like news corpora to produce longer-term timelines about individual people or organizations [34].

This work is concerned with how to effectively evaluate the quality of timeline items (sentences or text snippets) produced by such systems. Over the last few years new methodologies to evaluate the quality of timelines have been proposed [4, 27]. These methodologies typically use human annotators to manually identify atomic units of information that form a ground truth representing the information a 'good quality' summary about a topic should contain (see Figure 1). Next, textual items (e.g. sentences or tweets) returned by a diverse set of timeline generation systems for the topic are *pooled*. Finally, the pooled text items are manually judged to see what atomic information units for the topic they cover (if any), forming <text item,information unit> pairs. Metrics such as Expected Latency Gain [17] use the resultant pairs to estimate the degree to which individual text items included in a timeline (and hence the timeline as a whole) contains relevant, non-redundant, and timely information.

The use of atomic information units as a ground truth for evaluating timelines/real-time summaries is generally accepted and has been successfully deployed within the Temporal Summarization and Real-time Summarization tracks at the Text Retrieval Conference (TREC) [4, 27]. However, while these tracks produced test collections that can in theory be used to evaluate any timeline generation system, prior works have reported cases where these test collections fail to accurately estimate the performance of *new* timeline generation systems [32, 33]. In particular, it was observed in these past works that the overlap between items included in the initial pools (i.e. the assessed set) and those returned by their new proposed systems was insufficient to facilitate a robust comparison

of systems. As a result, it is unclear to what extent the test collections produced during these tracks can be used to evaluate the quality of new systems that were not pooled for judging [5].

This is an instance of a wider problem that is increasingly impacting information retrieval and information filtering tasks [56]. In particular, the offline datasets/data streams that are used to evaluate systems are rapidly increasing in size. However, the amount of assessment resources used to create the re-usable ground-truth component of such datasets/data streams by initiatives like TREC has not substantially increased over the last 10 years [9]. As a result, the relative proportion of datasets/data streams that are judged by assessors has dropped dramatically, which we argue that it is leading to increased metric instability when comparing systems. This is even more acute for tasks that require greedy algorithms to solve them, like timeline generation, as minor system variations can result in completely different outputs (e.g. due to the trickle-down effect from a decision made early on) leading to reduced output overlap between systems and thereby increasing assessment incompleteness when pooling. Hence, there is a need for better quantification of the impact of assessment incompleteness of datasets/data streams currently in use, as well as new means to mitigate metric instability resulting from this incompleteness [6, 15]. As such, in this paper we investigate to what extent current atomic information unit-based test collections are able to distinguish between timeline summarization systems with different effectiveness levels, as well as propose and evaluate automatic solutions to reduce the likelihood of errors occurring when evaluating such systems.

Contributions. The main contribution of our work is an in-depth analysis of the TREC 2013-2015 Temporal Summarization track test collections that quantifies how robust these collections are when evaluating systems not present in the assessment pool, as well as examine the effectiveness of different automatic <text item,information unit> expansion techniques aimed at increasing the robustness of these test collections. Specifically, we tackle the following four main research questions:

- **RQ1:** To what extent can the TREC Temporal Summarization track test collections accurately rank unpooled systems?
- **RQ2:** If we use automatic methods to generate additional <text item,information unit> pairs can we reduce the likelihood of new systems being mis-ranked?
- **RQ3:** What are the key properties of an effective system for generating <text item,information unit> pairs?
- **RQ4:** Given that automatic expansion can generate <text item,information unit> pairs that have not previously been assessed, what proportion of these 'unknown' matches are correct?

Our results show that the TREC 2013-2015 Temporal Summarization track test collections do not accurately estimate the effectiveness of unpooled systems. Moreover, the discrepancies observed between actual and estimated performances are sufficient to cause errors when ranking those systems. Furthermore, we found that the likelihood of encountering ranking errors is not uniform across system effectiveness levels – the better a system is, the more negatively it is impacted by not being pooled. For this reason, we conclude that it is potentially risky to use the TREC-TS test collections out-of-the-box. We then defined a three component framework (comprised of a Linking Strategy, Similarity Metric and Item Set) for automatically generating new <text item,information unit> pairs (matches), aimed at reducing these discrepancies. Our experiments using a range of instantiations of this framework show that automatically adding even a small number of <text item,information unit> pairs can markedly reduce the number of ranking errors observed when using the test collections. In particular, we found that the best framework configurations can reduce ranking errors by between 30% to 50%, and improve correlation against the correct ranking of systems by a statistically significant margin. Furthermore, by analyzing the matches produced by

different framework instantiations, we showed that to improve test collection robustness, accuracy when linking items to nuggets is crucial (as we might expect), but that surprisingly only a small number of added matches are needed to enhance test collection robustness (the best instantiations tested only exhibited around 15% match expansion recall). Finally, through a crowdsourced study examining previously unassessed matches produced by the framework, we observed that instantiations that used Shingle-based similarity were very effective at generating large volumes of correct matchings that were not in the ground truth. This indicates that broader automatic expansions of the TREC-TS test collections are viable, although performing such expansion needs to be undertaken with care. Overall, we conclude that automatic expansion techniques can improve the robustness of the TREC-TS test collections, reducing the risk of mis-ranking new systems that were not pooled.

The remainder of this paper is structured as follows. In Section 2, we provide a brief overview of timeline summarization and how it is evaluated, as well as relevant works that examine test collection robustness and how to enhance it. Section 3 defines 'robustness' in the context of timeline summarization systems and describes the experimental setup that we use throughout this paper. In Section 4, we examine RQ1, i.e. to what extent can the TREC Temporal Summarization track test collections accurately rank unpooled systems. Section 5 introduces our framework for automatically generating new <text item,information unit> pairs and investigates RQ2, i.e. can we use this framework to increase test collection robustness. In Section 6, we examine RQ3 by analyzing the matches produced by the framework with the aim of identifying the key properties of the expansions that were shown to be effective. Finally, Section 7 introduces a crowdsourced study into the large numbers of previously unassessed matches that the automatic expansion approach produce, with the aim of answering RQ4, i.e. are these matches predominantly correct or not. Conclusions are summarised in Section 8 and we provide a new expanded version of the TREC-TS test collection for researchers and developers in Section 9.

2 BACKGROUND AND RELATED WORK

In this section we introduce the background literature relevant to the remainder of this paper. In particular, we first provide a brief introduction into classical and timeline summarization, along with associated evaluation strategies in Sections 2.1 and 2.2. We then introduce the TREC Temporal Summarization (TREC-TS) track that forms the core of our investigation in this paper in Section 2.3. In Section 2.4 we discuss issues encountered with the TREC-TS test collections by prior researchers, meanwhile we discuss the issue and identify the knowledge gap in Section 2.5. Finally, we describe related works that attempt to solve similar issues and which we build upon in Section 2.6.

2.1 Classical Summarization Evaluation

In the summarization domain, a range of evaluation methodologies have previously been proposed and examined in the literature. Early works focused on estimating the quality of fixed-length textual summaries produced by either single-document or multi-document summarization systems [36]. This is a type of textual comparative evaluation, where a summary produced by an automatic system is compared against one or more gold standard summaries authored by humans. The idea underpinning this type of evaluation is that good summaries will be textually similar to the gold-standard summaries. To perform the similarity comparison, the ROUGE [24] suite of metrics have become the defacto standard and were used extensively as part of the Document Understanding Conference (DUC) [13] and Text Analysis Conference (TAC) [14] evaluations.

2.2 Timeline Summaries and Evaluation

Comparative evaluation approaches were used for many years to evaluate multi-document summarization systems [1], however the shift toward real-time information sharing and the associated development of timeline generation and real-time summarization solutions [32, 45, 51, 55] that push updates to users over an extended period of time required new evaluation methodologies. A timeline summary can be defined as a number of (approximately) sentence-length timestamped text items. These text items might be sentences extracted from news articles [4] or tweets [27]. A timeline summary is usually about a topic or event, and hence the text items it contains should be relevant to that topic or event. A timeline is normally visualized as a list of text items in chronological or reverse-chronological order. New text items may be added to the timeline over time, as new information emerges and is found by the summarization system. Classical comparative evaluation approaches that use metrics like ROUGE [24] and its temporal extensions [12, 17] make the assumption that both the summary to be evaluated and the gold-standard summaries are of (roughly) equal length, and that the gold-standard summaries do not change over time. As such, these classical comparative summary evaluation approaches are unsuitable to evaluate timelines.

To solve this issue, *atomic information units* were introduced as an alternative means to evaluate the quality of a timeline summary [17]. Atomic information units had been used in a wide range of domains prior to their application to timeline evaluation, such as Web search diversification [43] and question answering [48], although the terminology used to describe them changes depending on the domain they are applied to. Indeed, atomic information units are equivalent to sub-topics, aspects, facets, clusters or nuggets [17, 32, 37, 45, 54]. The core requirement behind atomic information unit-based evaluations is that all of the units that contribute to the evaluation score for a system should be explicitly defined. In this way, evaluation can be reduced to counting the proportion of all units covered by a system. The more units covered (typically within some range constraint such as the top k documents), the better that system is. This concept maps naturally into a summarization context, where each 'unit' represents a piece of information that 'good' timeline summary for a topic should contain.

In practice, for evaluating timeline summaries, atomic information units have been implemented in two different manners. First in the form of information nuggets within the TREC Temporal Summarization track during 2013 to 2015. Second as information clusters within the TREC Real-time Summarization track during 2016 and 2017. We choose to use the TREC Temporal Summarization implementation as the basis for the study in this paper as it is the more complex/costly to deploy of the two (due to the more fine-grained definition of atomic information units used) and because it enables a more detailed comparison of systems [5]. We discuss this implementaton below. For those interested in differences between the two tracks we recommend the comparison by [5].

2.3 TREC Temporal Summarization Track

In 2013 the Text Retrieval Conference (TREC) introduced the Temporal Summarization (TREC-TS) track that examined how to extract sentences from high volume streams of news and social content to return to the user as updates for large events [4]. TREC-TS is a timeline generation task, as defined above, where each topic is an event (represented by an event query, e.g 'costa concordia disaster'), the text items are sentences extracted from a stream containing news articles, blogs and other Web documents. To avoid differences in what might be considered a 'sentence', each document in the stream was pre-segmented. For a set of events, TREC-TS systems processed the high volume stream of sentences and emitted a subset of those sentences into a timeline summary.

For evaluation, TREC-TS adopted an atomic information unit-based evaluation methodology, where an information unit was referred to as a 'nugget'. This methodology was inspired by earlier work developed for question answering [48] and applied in a series of evaluations in the early 2000s.

Table 2. Statistics of the TREC Temporal Summarization test collections from 2013, 2014 and 2015.

	Year		
Statistic	2013	2014	2015
Number of Events	9	15	21
Number of Nuggets	1,168	1,394	996
Number of Matches	5,071	13,635	24,823
Number of Sentences	10,377	14,652	33,483

Nuggets in the TREC-TS context represented atomic facts relevant to an event, represented by short natural language phrases. For example, for the event 'Costa Concordia shipping disaster', the ground truth might contain nuggets such as 'occurred on Friday 13th January 2012', 'ran aground on a reef' and 'the hull was punctured'. Under this evaluation methodology a perfect summary is one that covers all of the information nuggets for an event while being as short (contains as few sentences) as possible. Summaries containing redundant (repeated) information are penalized and were also evaluated in terms of timeliness (was the information relevant at the time it was retrieved?).

As a TREC track dedicated to supporting standardized evaluation, TREC-TS produced three test collections for evaluating timeline generation systems, one for each of the years that the track ran (2013, 2014 and 2015). These test collections each contain a number of topics (events), a high-volume stream of sentences for each topic, and a ground truth label set comprised of the information nuggets along with a <text item,information unit> (i.e. <sentence,nugget>) mapping that describes what sentences contain the information represented by each nugget. Creating the ground truth label set for each test collection was a three-step process [2, 3]:

(1) **Nugget Extraction ('nuggetization')**: Human assessors manually defined the information nuggets for each topic. This was achieved by having TREC assessors read the edit stream from the Wikipedia page for each topic (the page describing the event). The assessors defined new information nuggets as they encountered novel information about the event that they considered important enough to be included in a "good" summary about the topic.
(2) **Sentence Pooling**: Each participating system submitted a timeline summary comprised of sentences for each topic (event). The systems assign each sentence a priority score indicating how confident they are that those sentences are of high-quality. The top-k sentences by priority score were then selected and added into a pool to be assessed.
(3) **Nugget Matching**: Given the ground truth nuggets extracted from Wikipedia, assessors then manually checked each sentence in the pool, recording whether those sentences contained any of the information represented by the nuggets. A sentence that contains a nugget's information is referred to as 'covering' or 'matching' that nugget. The result of this is a set of <sentence,nugget> pairs, specifying which sentences contain the information represented by each nugget. It is worth noting that nuggets represent concepts, hence the matching process often requires the assessor to do more than match the text of a concept to the text of a sentence, e.g. accounting for synonyms.

The statistics of the resultant TREC-TS test collections for each year are provided in Table 2. Both the nugget extraction and nugget matching steps involved significant human effort (by NIST assessors). According to a study by Baruah et al. [5], the total assessment time spent to create the 2013 and 2014 TREC-TS test collections was around 375 hours, where over 80% of that time was spent on nugget matching.

2.4 On the Robustness of TREC-TS Evaluation

The test collections produced by TREC-TS have been used for a range of research papers since their original release [5, 6, 19, 20, 32, 33]. However, a number of these works reported needing to add more nuggets/matches to the provided ground-truth sets to make the test collections usable. In particular, McCreadie et al. [32] reported in their paper that there was very low overlap between the sentences included in the TREC-TS pool and the top sentences selected by their system, i.e. assessment completeness [9] was low. To tackle this, they performed additional pooling and matching based on the TREC-TS guidelines, adding 22,424 sentences to the pool at a significant cost. This was then echoed in their later study [33] where they found almost no overlap between their diversification-focused system and the TREC-TS pool (see Annex A from [33] for details), again requiring the pooling and assessment of the new summaries. On the other hand, Ekstrand-Abueg et al. [15] performed a correlation study examining whether removing individual systems from the pool adversely affected the system ranking under the official track metrics. They reported high correlations between system rankings pre and post pooling, indicating that the test collections are reusable. However, they also noted that there were outlier systems that were severely affected (i.e. were mis-ranked) when they were removed from the pool.

These prior studies lead us to question to what extent the TREC-TS test collections are in fact robust when evaluating systems not included in the pool. The studies reported in [33] and [32] required significant additional pooling and assessment effort before the collections could be used. Having to perform reassessment for each new system or summary to be evaluated reduces the value that these test collections bring to IR evaluation. Hence, in this paper, we quantify how robust these collections are for evaluating unpooled systems and also propose and evaluate automatic techniques aimed at increasing the robustness of these test collections.

2.5 Incompleteness of Relevance Judgments

Apart from the initial examinations by [6, 15], the robustness of timeline generation test collections has not been explored in the literature. However, there have been a number of past works in the wider information retrieval domain (typically for search tasks) examining the effect that relevance assessments (or lack thereof) has on test collection robustness. Collections where not all items are judged are known as 'incomplete' collections, while the degree of 'incompleteness' of a test collection refers to the proportion of items that are judged (usually with respect to a pool of top-ranked items from automatic systems) [47]. For instance, early work by Voorhees [47] investigated how different relevance assessment sets for a test collection impacted on the evaluation of retrieval results for the TREC-4 and TREC-6 test collections. That study showed that while the effectiveness metrics were impacted by using assessments created by different groups (e.g. NIST assessors vs. Waterloo assessors), the resultant ranking of the retrieval runs (systems) were highly correlated. Meanwhile, Zobel [56], examined the fairness of top k pooling methods for selecting documents to assess, showing that a pooling depth of 100 appeared to be adequate for search over the TREC-5 test collection. These early studies support the idea that smaller collections are indeed robust in the face of incomplete assessments.

However, over time, the size of test collections used by evaluation campaigns like TREC grew, but the pool depth (the number of judged documents per topic) across years has remained roughly constant (due to the cost of human assessment), increasing the relative degree of incompleteness (e.g. due to the varied nature of documents retrieved by systems contributing to the pools for these large corpora). Hence, later studies such as that by Buckley and Voorhees [9] examined the effect that further relaxing the completeness assumption has on the Cranfield evaluation methodology in larger test collections. In contrast, they showed that the Cranfield methodology was not robust in the face of massively incomplete relevance judgments. In an early attempt to address these concerns,

Carterette and Allan [11] proposed an automatic expansion technique based on inter-document similarity and demonstrated that such a technique could be used to better evaluate retrieval systems. Meanwhile, works such as [42] have also questioned how robust different IR metrics are when using pooling at different k values. Parallels can be drawn between these works in the search domain and the questions investigated in this paper. The TREC-TS test collections are built on a corpus containing over a billion items (sentences).[1] However, only between 60 and 100 (depending on year [3, 4]) of the top k sentences were pooled from participating systems. Hence, assessment completeness is a valid concern when working with collections at this scale. For this reason, Baruah [6] performed a small study into augmenting the TREC-TS 2013 test collection by expanding the set of relevance assessments, through the incorporation of exact duplicates in order to obtain more accurate estimates of system performance. However, while the results from this approach appeared positive, they did not have a sufficient number of topics to confidently conclude that the approach was effective.

Although studies mentioned above have raised concerns over the effect of incomplete relevance assessments, none have explored automatic expansion of the relevance set beyond addition of exact duplicates in the limited context of TREC-TS 2013. Therefore, in this paper, we study a range of techniques for automatic expansion of the relevance set and analyze the effects of such expansions as generated by these methods on the robustness of TREC-TS 2013, 2014 and 2015 test collections.

2.6 Nugget-Based Expansion

To tackle the problem of automatic expansions within the context of TREC-TS, we need to determine if the information contained in a nugget is also contained within a given sentence. This problem has been previously studied in the domain of question answering [7, 25, 26, 28, 48–50] and prior works have also examined its application within the information retrieval domain [16, 30, 38–41] for the purposes of automatic expansion to address the challenges imposed by incomplete relevance assessments. A general assumption made by these approaches is that the relevance of a document is defined by presence of an "information nugget" relevant to the information need. For example, if the information need is "Find information about John Kennedy," and one of the potential information nuggets is "John Kennedy was elected president in 1960," then any document that contains this information nugget (in verbatim form or not) will be deemed relevant to the information need. Similarly, if the document does not contain any of the information nuggets then the document will be considered non-relevant. To perform matching between information nuggets and documents, a means to estimate similarity is needed. Prior works such as [40] have successfully employed shingle-based (sometimes referred to as n-grams based) matching. As such we build on this work later to perform automatic match expansion Section 5.

It should also be noted that with respect to the above domains there are two fundamental challenges that persist to date: (1) how do we obtain a complete set of information nuggets relevant to an information need?, and (2) how do we match these information nuggets to documents (or items) in order to infer their relevance? That being said, based on [16, 30, 38–41], focusing on improving the matching between nuggets and items has been shown to be the more promising direction in order to increase the robustness of test collections. Hence, why we focus on investigating techniques to perform automatic matching between sentences and nuggets in this paper.

[1] http://trec-kba.org/kba-stream-corpus-2014.shtml

3 METHODOLOGY AND SETUP

To examine the extent that the TREC-TS test collections are robust, we need a standardized setting and evaluation methodology with which to quantify 'robustness'. To create such a setting, we first define what we mean by 'robustness' below:

Robustness: In a timeline generation context, a truly robust test collection is one which can be used to accurately estimate the quality of a timeline summary, regardless of whether that summary was included within the initial pool or not. A robust test collection should correctly rank different timeline generation systems in order of the quality of the timeline summaries they produce.

Given this definition, to evaluate the robustness of a test collection we can identify three main requirements: 1) a series of systems that produce summaries of known quality (such that we have a known ordering of systems); 2) an evaluation metric that reflects the quality of a system according to the test collection; and 3) we need to have the ability to compare systems when included in the pool and when excluded from the pool. Below we discuss how we design our experimental setup to meet these three requirements.

3.1 Synthetic System Generation

The first requirement for our evaluation is to have a series of timeline generation systems that can produce timeline summaries of known quality. This is so that we have a gold standard ranking of systems that reflects their actual performance. Initially, one might consider using the systems originally submitted to the TREC track in each year. However, this has some notable limitations. First, the systems that participated in TREC are different from year-to-year and the sources for those systems are not always available, hence we cannot deploy each TREC system across all years. This is potentially problematic, as there are relatively few topics ('events') in each test collection (between 9 and 21, see Table 2), which is less than the recommended number of topics for an IR experiment [10]. Indeed, this limitation was previously encountered by the Baruah [6] in their study. Second, the TREC systems only represent a subset of the range of possible system performances, e.g. in the first year, all participating systems were rather poor in terms of effectiveness. It would be preferable to be able to deploy single set of systems across all years such that we can compare across a larger number of events and have those systems represent the full range of system effectiveness (poor to perfect).

To achieve this, we instead take an alternative approach inspired by prior work in the Web and expert search domains [29, 46], where we generate synthetic systems with known performances. This is possible, since we are using the TREC-TS test collections as the subject of our investigations, which have sentence-level labels that quantify how much value is added by any sentence. Hence, we can define a synthetic system that takes in the sentence-level labels along with a target effectiveness level, and generates summaries with (approximately) that effectiveness level for each topic.

In particular, as discussed in Section 2.3, the TREC test collections contain atomic information items (nuggets) that form a ground truth for measuring summary quality. More precisely, the test collections contain <sentence,nugget> pairs that specify which individual sentences contain the information represented by each nugget. Following the atomic information nugget-based evaluation paradigm, a simple way to represent the quality of a timeline summary with k sentences is to calculate the proportion of nuggets it covers. For example, if for an event we have 50 unique information nuggets, and our summary contains sentences that are matched to 28 of those nuggets, then we can say that the summary covers 28/50 nuggets, or has 56% coverage. As long as all summaries are of the same length k for a topic, then nugget coverage is a fair representation of timeline summary quality (it measures the volume of information contained in a fixed amount of space).

Given the above, we specify a series of target effectiveness levels in terms of nugget coverage from 95% to 5% in 5% increments. For each target effectiveness level, we generate one summary per topic within the three TREC-TS test collections. For a topic, we first randomly select a subset of the information nuggets that matches the target effectiveness level, e.g. for 60% coverage, we select 60% of the nuggets for that topic. We then iterate over all <sentence,nugget> pairs for that topic in the ground truth label set, selecting one sentence that matches each nugget in a greedy manner. For instance, if a topic contained 100 nuggets and our target effectiveness level was 40%, we would first randomly select 40 of those 100 nuggets, and then attempt to select one sentence matching each of those 40 nuggets. When considering timeline summarization systems, not all sentences are equally likely to be selected (some are easier to find than others, e.g. because they contain the event query terms) and most nuggets have multiple sentences we might select. To capture this, instead of randomly selecting any of the available sentences that match the nugget, we instead use a probabilistic selection of sentences, based on the likelihood of each sentence having been selected by the original TREC systems (the more TREC systems that selected a sentence the more likely our synthetic systems will similarly select that sentence). As a sentence may cover multiple nuggets, we exclude a sentence from being selected if it covers any nuggets not in our target set. Furthermore, only around 70% of nuggets have associated matching sentences, i.e. in the remaining cases no systems in the pool found sentences that covered that nugget. In all cases we select as many sentences as possible and then 'fill' the remaining slots (to maintain a consistent length k) with redundant sentences that do not contain any relevant information. In practice, this means that the actual nugget coverage for a synthetic summary is lower than the target coverage, e.g. a 90% coverage target results in 68% actual coverage when averaged across topics. We summarize the statistics of our generated synthetic runs in Table 3. As can be observed from Table 3, this synthetic system generation approach produces a range of systems that span the range of effectiveness levels attainable in terms of nugget coverage.

3.2 Evaluation Metrics

Having produced a set of systems with known performances, we now need to define metrics to capture how effective the summaries produced by those systems are. One possible option would be to simply use nugget coverage averaged across topics as an estimation of summary quality, as we did in the previous section. However, the TREC-TS track also considered factors beyond nugget coverage, such as novelty, brevity and latency [17]. For this reason, as well as to maintain compatibility with the track, we use the official TREC-TS target evaluation metric, which itself is the harmonic mean between two metrics: Expected Latency Gain and Latency Comprehensiveness. Expected Latency Gain (ELG) is a precision-like metric, calculated as the sum of the relevance of each nugget that a sentence covered, computed as:

$$ELG(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{u \in \mathcal{S}} \sum_{n \in \mathbf{M}(u)} \mathbf{g}(u, n) \tag{1}$$

where $\mathcal{S}$ is the stream of sentences returned by the system, $\mathbf{M}(u)$ is the set of gold standard nuggets matching sentence u (as determined by an assessor) and $\mathbf{g}(u, n)$ measures the utility of matching sentence u with nugget n. Latency Comprehensiveness (LC) is the proportion of all nuggets matched by the system updates, computed as:

$$LC(\mathcal{S}) = \frac{1}{|\mathcal{N}|} \sum_{u \in \mathcal{S}} \sum_{n \in \mathbf{M}(u)} \mathbf{g}(u, n) \tag{2}$$

where $\mathcal{N}$ is the set of nuggets for the current event. For both ELG and LC, the $\mathbf{g}(u, n)$ component contains built-in penalties to capture sentence brevity and latency. We refer the reader to the

Table 3. Synthetic Run Statistics.

Synthetic System	Target Coverage	Actual Coverage	TREC-TS Metrics		
			ELG	*LC*	$\mathcal{H}$*(ELG,LC)*
Synth-C95	95%	72%	0.3590	0.6989	0.4589
Synth-C90	90%	68%	0.3337	0.6474	0.4249
Synth-C85	85%	64%	0.3336	0.6277	0.4216
Synth-C80	80%	61%	0.3404	0.5901	0.4165
Synth-C75	75%	57%	0.3241	0.5676	0.3996
Synth-C70	70%	53%	0.3202	0.5289	0.3834
Synth-C65	65%	49%	0.3189	0.5105	0.3792
Synth-C60	60%	46%	0.3057	0.4466	0.3488
Synth-C55	55%	41%	0.2922	0.4233	0.3314
Synth-C50	50%	38%	0.2997	0.4181	0.3371
Synth-C45	45%	34%	0.2888	0.3596	0.3047
Synth-C40	40%	30%	0.2919	0.3398	0.3002
Synth-C35	35%	25%	0.2989	0.2961	0.2824
Synth-C30	30%	22%	0.2737	0.2515	0.2501
Synth-C25	25%	18%	0.2555	0.2012	0.2108
Synth-C20	20%	14%	0.2785	0.1523	0.1869
Synth-C15	15%	10%	0.3053	0.1199	0.1623
Synth-C10	10%	7%	0.3122	0.0755	0.1121
Synth-C05	5%	3%	0.2478	0.0274	0.0473

TREC track metrics documentation[2] for a detailed explanation on how these are calculated. To provide a target metric, an *F*-like measure was also defined, which we denote $\mathcal{H}$*(ELG,LC)*. This is the harmonic mean of ELG and LC,

$$\mathcal{H}(ELG, LC)(\mathcal{S}) = 2 * \frac{ELG(\mathcal{S}) * LC(\mathcal{S})}{ELG(\mathcal{S}) + LC(\mathcal{S})} \tag{3}$$

We report the performance of our synthetic systems under the TREC-TS Metrics *ELG*, *LC* and $\mathcal{H}$*(ELG,LC)* in Table 3. As we can see, the performance as reported by the TREC-TS *LC* and $\mathcal{H}$*(ELG,LC)* and metrics are highly correlated with the actual nugget coverage of the systems.

3.3 Depooling Methodology

Finally, to evaluate the robustness of the test collections, we need to be able to evaluate the difference in performance of systems when they are included within the pool and when excluded from it (we refer to this as being *depooled*). The core idea is that if a test collection is robust, then the estimated performance (under $\mathcal{H}$*(ELG,LC)*) of a pooled system with known coverage X should be similar to the estimated performance for that same system when it is not pooled. In this case, a system that is excluded from the pool represents a hypothetical new system that did not participate in the original TREC track and hence was not pooled (and as such would not have been considered when selecting items for judging).

TREC-TS followed a top k pooling methodology, where the sentences with the k highest confidence scores were added to the pool and later assessed (i.e. they took part in the nugget matching phase resulting in the <sentence,nugget> pairs $\mathbf{M}(u)$). From the TREC-TS pool statistics, we know the number of the original TREC-TS participating systems that contributed each sentence. We refer

[2]http://www.trec-ts.org/metrics-10242013.pdf

Table 4. Synthetic Run Performances under $\mathcal{H}$*(ELG,LC)* when pooled or depooled. ▼ indicates statistically significant decreases in estimated performance (t-test p<0.01) between the run when in the pooled and when depooled.

Synthetic System	$\mathcal{H}$*(ELG,LC)* When Pooled	$\mathcal{H}$*(ELG,LC)* When Depooled
Synth-C90	0.4249	0.3850▼
Synth-C80	0.4165	0.3823▼
Synth-C70	0.3834	0.3480▼
Synth-C60	0.3488	0.3196▼
Synth-C50	0.3371	0.3105▼
Synth-C40	0.3002	0.2617▼
Synth-C30	0.2501	0.2259▼
Synth-C20	0.1869	0.1627▼
Synth-C10	0.1121	0.0687▼

to sentences contributed by multiple systems as *common sentences* and sentences that were only contributed by a single system as *uncommon sentences*.

Building on past work examining the effect of not including systems when pooling on IR system performance [15], we simulate the state of the TREC-TS test collections in scenarios where a particular system was not pooled. For ease of reference, we refer to this as *depooling*. Depooling involves removing one copy of each of the top k sentences contributed by that system from the pool, along with any associated <sentence,nugget> pairs that resulted from the subsequent nugget matching phase. By definition, common sentences would not be affected by removing only a single system (that system's sentences would still be contributed by some other system). However, uncommon sentences would be at risk from being eliminated from the pool entirely. If a sentence is eliminated from the pool, then that loss will impact the scoring of all systems. As we used the sentences in the TREC-TS pool previously to produce our synthetic systems, those systems behave as though they have been pooled. Hence, by depooling one of the synthetic systems we can investigate whether its estimated performance would have been adversely impacted had it not been pooled (i.e. due to its uncommon sentences not being assessed). As such, we create an evaluation scenario for each of our 19 systems, each representing the case where that system was depooled. In the next section, we use these depooling scenarios to answer our first research question, i.e. RQ1 'To what extent can the TREC Temporal Summarization track test collections accurately rank unpooled systems?'.

4 RQ1: TO WHAT EXTENT ARE THE TREC-TS TEST COLLECTIONS ROBUST?

To answer RQ1, we first examine whether the estimated performance scores for systems change when pooled and when depooled. The ideal outcome is that the scores would not change, however this would only occur in cases where the system being depooled was totally comprised of common sentences. Hence, we can expect some score variance due to uncommon sentences being eliminated from the pool, but we would hope such score variance is minimal. Table 4 reports the estimated performance of the synthetic systems under $\mathcal{H}$*(ELG,LC)* in the pooled and depooled scenarios (for brevity we only list performances for half the systems, the observations are the same for the other systems). As we can observe from Table 4, in all scenarios, depooling a synthetic system causes a statistically significant decrease in its estimated performance under the official TREC metric ($\mathcal{H}$*(ELG,LC)*). This is a first indication that the test collections may not be as robust as we would like, as the effectiveness scores estimated for a system is shown to vary greatly depending

Table 5. Effect of depooling a single system in terms of ranking stability.

		Rankings Pooled Vs. Depooled	
Synthetic System	# Rank Swaps	Kendall's τ	τ_{AP}
Synth-C95	3	0.9649	0.8129
Synth-C90	2	0.9766	0.8752
Synth-C85	6	0.9298	0.8771
Synth-C80	0	1.0000	1.0000
Synth-C75	4	0.9532	0.9081
Synth-C70	1	0.9883	0.9914
Synth-C65	3	0.9649	0.9579
Synth-C60	1	0.9883	0.9915
Synth-C55	4	0.9532	0.9614
Synth-C50	1	0.9883	0.9914
Synth-C45	2	0.9766	0.9864
Synth-C40	1	0.9883	0.9870
Synth-C35	1	0.9883	0.9946
Synth-C30	0	1.0000	1.0000
Synth-C25	1	0.9883	0.9006
Synth-C20	0	1.0000	1.0000
Synth-C15	0	1.0000	1.0000
Synth-C10	0	1.0000	1.0000
Synth-C05	0	1.0000	1.0000
Average	1.5790	0.9815	0.9597

on whether it was included in the pool or not. Hence, it is likely that *new* systems that were not originally pooled will have their true performance underestimated.

On the other hand, some error when estimating the performance of depooled systems is to be expected, as this is a known issue with pooling-based evaluation scenarios [9, 15]. From an evaluation perspective, what researchers and developers care about is whether the test collection is able to distinguish between systems with different effectiveness levels, i.e. whether we get the ordering of systems correct (particularly in the top ranks) is more important than whether individual system scores are underestimated [5]. Indeed, while we might expect that underestimations of a system's performance will cause that system to be mis-ranked, evidence from the search domain indicates that IR metrics tend to have some degree of robustness against incompleteness effects [42], i.e. the error in the score for a system may not be sufficient to cause a ranking swap.

As our synthetic systems have known effectiveness levels (based on nugget coverage), we know the correct ordering of systems. We also showed previously in Table 3 that the official TREC-TS metric ($\mathcal{H}$*(ELG,LC)*) reflects this correct ranking when all systems are pooled. However, given that we know that depooling a system causes statistically significant changes in $\mathcal{H}$*(ELG,LC)*, it is possible that these changes are severe enough to result in that system (and other systems) being mis-ranked. In Table 5, we report the effect that depooling each system individually has on the overall ranking of synthetic systems in terms of number of rank swaps (mis-rankings) and overall rank correlation under Kendall's τ. Additionally, as we are often more interested in distinguishing between systems near the top of the ranking than those at the bottom, we also report τ_{AP} [52] values, which place higher weight on rank correlations occurring in the top ranks. Ideally, we should preserve the original correct ranking, i.e. the number of rank swaps should be 0, while Kendall's τ and τ_{AP} would

be 1. From Table 5, we can see that for the majority of scenarios, depooling a single system results in the mis-ranking of at least one pair of systems. Indeed, the average number of system swaps needed to restore the correct ranking across depooling scenarios is 1.579, while the rank correlation on average was around 0.9815. This result is similar to that reported by Ekstrand-Abueg et al. [15], who observed correlation values around 0.97 when holding out individual TREC-TS systems in their study. However, while these rank correlations appear high, it is important to remember that systems are still being ranked incorrectly. Moreover, from Table 5, we see that rank swaps are more common when highly effective systems are depooled, i.e. if you have a new (unpooled) system that is very effective, it is likely to be mis-ranked (see the top of Table 5). On the other hand, it appears that systems at the lower end of the effectiveness scale have little impact on the overall ranking of systems when not pooled.

Summary: To answer RQ1, we conclude that the TREC-TS test collections are likely robust when evaluating systems that were not pooled at the lower end of the effectiveness scale, i.e. systems equivalent to or worse than Synth-C30, that has an actual nugget coverage level of 22%. On the other hand, systems that were not pooled that push the upper-end of the effectiveness envelope are more likely to be miss-ranked, and hence using the TREC-TS test collections out-of-the-box is subject to more risk. The issue is that a researcher or developer has no way of knowing which case they fall into. As such, it would be advantageous to improve the test collections to reduce the risk of ranking error for unpooled/depooled systems.

5 RQ2: CAN WE USE AUTOMATIC MATCHING TO INCREASE ROBUSTNESS?

In the previous section we observed that systems that are depooled (i.e. representing new systems that did not participate in the original TREC tracks) are at risk from being mis-ranked. In particular, when comparing the ranking of the same 19 systems when all were pooled vs. when only 18 of them were pooled, we observed that ranking errors start to occur (average Kendall's τ and τ_{AP} values of 0.9815 and 0.9597, respectively).

These ranking errors stem from a system identifying sentences that are: 1) highly important (e.g. they cover a nugget that is very difficult to find sentences for), 2) uncommon (no other system contributed those sentences to the pool) and 3) the system assigned them a high confidence score (so they would have been added to the pool if the system had been part of the initial evaluation). The result of such a system not being included in the pool is that a portion of its top k documents will not have been assessed,[3] and unassessed sentences are assumed to not be relevant to any nuggets. Hence, that system's performance estimation is likely to be an underestimate.

From a test collection perspective, such a system not being included in the pool leads to missing <sentence,nugget> pairs. These pairs could be recovered by pooling all new systems and then re-assessing, however, this additional cost eliminates much of the value that these test collections bring to IR evaluation. Indeed, if we use the total amount of time it reportedly took the TREC assessors to perform nugget matching [5] then each additional sentence assessed takes around 50 seconds on average. Moreover, this does not factor in time taken to set up the assessment system and recruit assessors. As such, it would be advantageous to have an automatic means to generate the missing <sentence,nugget> pairs without resorting to more human assessment.

In the remainder of this section we examine methods for automatically generating missing <sentence,nugget> pairs using the initial set of <sentence,nugget> pairs from the TREC-TS pool as a base, which we refer to as *match expansion*. In particular, we first discuss our experimental methodology (Section 5.1) as well as evaluation metrics (Section 5.2). Then we introduce a framework for performing automatic match expansion comprised of three components (i.e. a Linking Strategy,

[3]Recall that the top k pooling methodology guarantees that all pooled systems will have had their top k documents assessed.

Similarity Metric and Item Set) to generate new <sentence,nugget> pairs (Section 5.3). Finally, we report our experimental results in Section 5.4.

5.1 Matching Expansion Methodology

To evaluate matching expansion, we need two sets of sentences for which we know what the correct <sentence,nugget> pairs are. The first set we need represents the sentence pool pre-expansion, while the second set represents the sentences and <sentence,nugget> pairs that are the correct expansions our proposed system should produce. Previously in Section 3.3 we created such a setting via depooling, which we re-use here. In particular, for each of our synthetic systems, by depooling that system we create a scenario where some sentences and associated <sentence,nugget> pairs will be eliminated from the pool. The goal of a matching expansion algorithm in this case is to restore as many of those sentences and associated <sentence,nugget> pairs as possible, while avoiding introducing erroneous <sentence,nugget> pairs.

5.2 Robustness Metrics

Recalling our definition of *robustness* provided in Section 3, we consider an evaluation robust if it can accurately estimate the quality of a timeline summary. As we are primarily interested in comparing different systems, this can be operationalized by comparing the ranking of systems after expansion has been performed to the known correct ranking of systems (recall that we crafted synthetic systems with known performances such that we know what the correct ranking of systems is). Based on the above methodology, we have multiple 'scenarios', where in each we depool a single system and then apply an automatic expansion technique. As we have 19 systems (Synth-C95 to Synth-C05), we have 19 such scenarios. For each scenario we compare the ranking of systems post-expansion to the correct ranking of systems. We average performance across the 19 scenarios to form our final scores. We report three different metrics that capture the similarity between the correct system ranking and the ranking after expansion:

- **Avg Rank Swaps**: This is the raw number of pair-wise swaps needed to restore the correct ranking of systems when starting from the ranking post-expansion. This count is averaged across the 19 scenarios. A lower number of rank swaps is better.
- **Avg. τ ($a\tau$)**: This is the Kendall's Correlation [44] between the correct ranking of systems and the ranking of systems post-expansion. This correlation is averaged across the 19 scenarios. A higher correlation with respect to the correct ranking is better.
- **Avg. τ_{AP} ($a\tau_{AP}$)**: This is AP rank correlation coefficient [53] between the correct ranking of systems and the ranking of systems post-expansion. This is similar to $a\tau$, with the exception that it is 'top-heavy', i.e. it cares more about correlation at the top of the ranking. This is useful as we are often more interested in being able to distinguish good (high-ranked) systems than poor (low-ranked) systems. This correlation is averaged across the 19 scenarios. A higher correlation with respect to the correct ranking is better.

5.3 Matching Expansion Framework

Having defined how we can measure the impact of automatically expanding the ground truth, we next describe how we perform the expansion itself. In particular, when performing expansion there are three components that we need to consider, namely: how do we identify new matches (i.e. what *Linking Strategy* do we use); how do we measure similarity between items and/or nuggets (i.e. the *Similarity Metric*); and finally which items (sentences) from the corpus do we consider as valid candidates for expansion (we refer to this as the *Item Set*). Different approaches can be taken for each of these three components, which will impact on the quality of the final expansion. Hence, we

group these three components into a common expansion framework, where we first select a item set to form the basis of expansion, apply a similarity metric to find similar sentences/nuggets for each of the initial sentences (items), and finally use a linking strategy to infer new <sentence,nugget> pairs for the similar sentences. We experiment with different instantiations of each component as described below:

Item-Set: First, it is worth considering which sentences to use when performing expansion. In particular, there are two types of expansion that we consider here, referred to as *Relevant* and *All*. In the *Relevant* case, we only allow expansion for sentences that we already know are relevant, i.e. they have one or more existing matches. We expect this type of expansion to be high precision, but low recall, as we won't add any previously unjudged sentences into the ground truth, instead only adding more matches for those that are already included. In the *All* case, we allow expansion using any sentence in the document stream for each event. This will allow for much greater expansion, but is more likely to add noise to the test collection.

Linking Strategy: There are two different ways one might attempt to generate expansions for <sentence,nugget> pairs, which we experiment with in the remainder of this paper:

- *Item-Item*: First, given a $< A, n_i >$ pair, we can try to find other sentences that are similar to A, based on the assumption that if two sentences are similar, they will match the same nuggets. Then for each sentence B that is similar to A, we add a $< B, n_i >$ pair along side the existing $< A, n_i >$ pair. We refer to this as *Item-Item linking*.
- *Nugget-Item*: Second, we can start from a nugget n_i (which has a textual description) and try and find sentences that are similar to the description for n_i. If the sentence C is similar to the n_i's description, then we add a $< C, n_i >$ pair. We refer to this as *Nugget-Item linking*.

Similarity Metric: For both linking strategies we need to define 'similarity' between texts, i.e. sentence-to-sentence similarity in the case of Item-Item linking, or nugget description-to-sentence similarity in the case of Nugget-Item linking. In our later experiments we experiment with three different types of text similarity:

- *Levenshtein Distance*: For this expansion method, we take the simplest of approaches, using edit distance between the two strings with Levenshtein distance [18].
- *Shingle-Based Similarity:* This expansion method employs a text-based matching algorithm that is used in [41] in order to automatically infer the relevance of documents given the nuggets extracted. The matching algorithm is based on a variant of shingle matching, which is often used in near-duplicate detection [8]. A shingle is a sequence of l consecutive words in a piece of text. For example, after stopwording, the nugget *"John Kennedy was elected president in 1960"* has the following shingles for l = 3: *(John Kennedy elected), (Kennedy elected president)*, and *(elected president 1960)*. A similarity score is obtained by (1) computing a score for each shingle, and (2) combining these shingle scores to obtain a score for each nugget. For any nugget n and each shingle s of size l, let *S(D,n)* be the minimum span of words in the document D that contains all shingle words in any order. A shingle is similar if it is contained in a small span. We used the algorithm presented in [23] to find the shortest span of shingle words in a text document in linear time. Note that in contrast to standard shingle matching, we do not require all shingle words to be present in the matching document in the same order or contiguously; by our definition, such a shingle would have a "perfect" shingle score of 1. We define the shingle score as follows, where λ is a fixed decay parameter[4]:

[4]We have previously found $\lambda = 0.8$ to be an effective value and hence use it throughout our experiments.

$$\text{ShingleSimilarity(D,n)} = \frac{1}{|Shingles(n)|} \sum_{s \in Shingles(n)} \lambda^{(S(D,s)-l)/l} \quad (4)$$

- *Semantic Similarity:* As sentences in the TREC-TS test collections are drawn from an article stream comprised of news articles, blogs and forum posts, we can expect significant vocabulary miss-match between different articles discussing the same information. For this reason, it is reasonable to expect that raw text similarity will fail to identify some similar sentences due to vocabulary miss-matches. To tackle this issue, we also experiment with the identification of similar sentences based on semantic rather than textual similarity. In this case, we represent each sentence as a high-dimensional sentence embedding and then calculate similarity in terms of that semantic space. Sentence embeddings have previously been shown to be an effective sentence representation when calculating similarity [21]. In particular, given a sentence, for each word in that sentence, we first convert that word into a high-dimensional word embedding. To produce a sentence embedding we calculate a single position per dimension by averaging the word positions per dimension. For reproducibility, we use Word2Vec [35] along with pre-trained word embeddings from the Google News dataset (about 100 billion words).[5] We use Cosine similarity for calculating the distance between the sentence embedding vectors.

For all three similarity metrics we need to define a similarity threshold, above which a nugget and sentence will be considered a match. The correct similarity threshold will vary between techniques, e.g. what might be considered an acceptable threshold for raw text similarity will differ from semantic similarity. For brevity, in the following section we only report performances from around the peak threshold θ observed based on experimentation with different θ ranges.

5.4 Expansion Robustness Results

Table 6 reports the effectiveness of our proposed expansion techniques in terms of the metrics discussed in Section 5.2. Under the robustness metrics, # Rank Swaps is the number of swaps needed on average to re-create the correct ranking (lower is better), while $a\tau$ and $a\tau_{AP}$ indicate the resultant correlation between the ranking produced post-expansion and the correct ranking (higher is better). In the case of the Ranking Metrics $a\tau$ and $a\tau_{AP}$, we also report statistically significant changes (paired t-test $p < 0.05$) against no expansion (None). To structure our discussion of the results, we divide our analysis below based on the linking approach and item set used, forming three categories: Relevant/Item-Item, All/Item-Item and All/Nugget-Item.

Relevant/Item-Item: To begin our analysis, we examine the performance when expanding using the 'Relevant' item-set. Recall that this is a form of very conservative expansion, where only sentences that we already know are relevant are used as the basis for expansion. Rows 2-11 of Table 6 report the performance of relevant item set expansion (item-item with either levenshtein or semantic similarity). If we examine the effect that using item-item expansion with levenshtein similarity (Relevant/Item-Item/Levenshtein) has on the ranking of systems, we see that this type of expansion results in fewer ranking errors than observed prior to expansion (None). Indeed, the average number of swaps needed to restore the correct ranking drops by 30% from 1.5789 to 1.0526 (threshold=0.9) and average Kendall's τ correlation against the correct ranking increases by a small but statistically significant margin (0.9815 to 0.9877) across the scenarios tested. Importantly this shows that the restoration of a relatively small proportion of all sentences that should have been

[5] Available from https://code.google.com/archive/p/word2vec/

Table 6. Correlation with the correct system ranking on average across depooling scenarios and topics when performing item-to-item and nugget-to-item similarity expansion. Statistically significant changes (paired t-test p<0.05) in $a\tau$ and $a\tau_{AP}$ against no expansion (None) are denoted $\triangle$ and $\triangledown$ for increases and decreases respectively.

Expansion Technique				Ranking Metrics		
Item-Set	Linking	Similarity	Threshold θ	Avg Rank Swaps	$a\tau$	$a\tau_{AP}$
None			-	1.5789	0.9815	0.9597
Relevant	Item-Item	Levenshtein	0.99	1.4211	0.9834	0.9597
Relevant	Item-Item	Levenshtein	0.90	1.0526	0.9877$\triangle$	0.9619
Relevant	Item-Item	Levenshtein	0.80	1.0526	0.9876$\triangle$	0.9610
Relevant	Item-Item	Levenshtein	0.70	1.0526	0.9876$\triangle$	0.9610
Relevant	Item-Item	Semantic	1.00	1.1579	0.9865	0.9625
Relevant	Item-Item	Semantic	0.98	0.8421	0.9902$\triangle$	0.9640
Relevant	Item-Item	Semantic	0.96	**0.6842**	**0.9920**$\triangle$	**0.9878**$\triangle$
Relevant	Item-Item	Semantic	0.94	1.9473	0.9772	0.9334
Relevant	Item-Item	Semantic	0.92	5.8421	0.9317$\triangledown$	0.8361$\triangledown$
Relevant	Item-Item	Semantic	0.90	8.2105	0.9040$\triangledown$	0.7359$\triangledown$
All	Item-Item	Shingle	1.00	1.3158	0.9846	0.9707
All	Item-Item	Shingle	0.90	2.2632	0.9735$\triangledown$	0.9611
All	Item-Item	Shingle	0.80	1.6316	0.9809	0.9218$\triangledown$
All	Item-Item	Shingle	0.70	1.8421	0.9785	0.9549
All	Item-Item	Shingle	0.60	1.2632	0.9852	0.9744
All	Item-Item	Shingle	0.50	1.3158	0.9846	0.9744
All	Item-Item	Semantic	1.00	1.0000	0.9883$\triangle$	0.9743
All	Item-Item	Semantic	0.98	2.0526	0.9760	0.9536
All	Item-Item	Semantic	0.96	1.5263	0.9821	0.9474
All	Item-Item	Semantic	0.94	8.1053	0.9052$\triangledown$	0.7874$\triangledown$
All	Nugget-Item	Shingle	1.00	1.4211	0.9834	0.9567
All	Nugget-Item	Shingle	0.90	1.4737	0.9828	0.9557
All	Nugget-Item	Shingle	0.80	1.4211	0.9828	0.9605
All	Nugget-Item	Shingle	0.70	1.4737	0.9834	0.9455
All	Nugget-Item	Shingle	0.60	1.3684	0.9840	0.9573
All	Nugget-Item	Shingle	0.50	1.4211	0.9834	0.9452
All	Nugget-Item	Semantic	1.00	1.5789	0.9815	0.9598
All	Nugget-Item	Semantic	0.98	0.8421	0.9902$\triangle$	0.9640
All	Nugget-Item	Semantic	0.96	**0.6842**	**0.9920**$\triangle$	**0.9878**$\triangle$
All	Nugget-Item	Semantic	0.94	1.9474	0.9772	0.9335

pooled (e.g. around 10%) can eliminate around 30% of the ranking errors (1.5789 rank swaps to 1.0526 rank swaps). Next, we examine how effective item-item expansion over the relevant item-set is when is when using semantic similarity rather than using levenshtein distance (Relevant/Item-Item/Semantic). From Table 6 we observe that when the vector similarity threshold θ is set to 0.98 and 0.96, we further reduce the average number of rank swaps (ranking errors) to 0.8421 and 0.6842, respectively. If we compare this to the number of ranking errors prior to expansion, then semantic expansion can reduce the number of errors by up to 50% (1.5789 rank swaps to 0.6842 rank swaps). This also increases the correlation between the system ranking post-expansion and the correct ranking by a statistically significant margin under both $a\tau$ and $a\tau_{AP}$.

All/Item-Item: Having shown that when performing conservative expansion using only sentences previously labeled as relevant we can decrease ranking error, we next examine how effective the expansion is when we broaden the set of sentences considered to any sentence in the stream. This can be considered a 'riskier' form of expansion, as we can now potentially introduce irrelevant sentences into the ground truth, but enables the identification of a much wider set of relevant sentences. Table 6 reports the impact of expanding using all sentences in rows 12-21. As before, we break this type of expansion based on the similarity metric. In this case, we experiment with both Shingle-based expansion and Semantic expansion. Starting with shingle-based expansion (All/Item-Item/Shingle), we see that it has little overall impact on the ranking of systems, e.g. when θ=0.5, correlation with the correct ranking ($a\tau$) only increases by a very small margin (0.0031). This is an interesting result, as with such a low threshold, we would expect a large number of expansions to be added, but those expansions do not seem to be impacting the ranking of systems significantly (we examine this in more detail in Section 6). Examining semantic expansion on the other hand (All/Item-Item/Semantic), we observe that rank correlation with the correct ranking is neutral or negatively impacted for the majority of thresholds θ. This is interesting, since we previously observed that this same expansion when applied over the Relevant set significantly improved robustness, but widening the sentence set used for expansion appears to eliminate most of the gain from this approach. For example, when θ=0.98, two rank swaps are needed on average to restore the correct ranking when using all sentences, in contrast to only 0.8421 rank swaps when only expanding with previously known relevant sentences. The problem becomes worse when using threshold θ=0.94, where 8.1053 rank swaps were required on average to restore the correct ranking when using all sentences, as compared to 1.9473 rank swaps when only expanding with the 'Relevant' item-set. This indicates that as we might expect, doing semantic expansion with any sentence is risky, since we are likely to find sentence pairs that are (semantically) similar but do not share the same information nuggets. For instance, consider the nugget '675+ injured'. Semantic expansion using word embeddings will identify a sentence such as:

"A packed commuter train slammed into a retaining wall at a railway terminus in Buenos Aires during rush hour Wednesday, leaving at least 49 dead, 550 injured, and dozens trapped in the wreckage."

as similar, since 'injured' is an exact match with 'injured' and '675+' is semantically similar to '550' in our embedding space, where shingle-based similarity would not. However, this is in fact an erroneous match, since in our nugget set we also have nugget '550 injured', hence semantic-based expansion would lead us to effectively conflate these two nuggets in our ground truth. On the other hand, there is one setting where Item-Item Semantic Expansion over all sentences is effective, i.e. when θ=1.0. Hence, we conclude that All/Item-Item/Semantic can be better than All/Item-Item/Shingle, but both are inferior to Relevant/Item-Item/Semantic.

All/Nugget-Item: Finally, we examine the performance of the alternative linking approach when generating new matches, i.e. Nugget-Item Expansion. Recall that in this scenario, instead of identifying similar sentences and then inferring matches, we directly match nuggets to sentences based on similarities between the text of the nugget and the text of each sentence. Table 6 reports the impact of expanding using all sentences for nugget-item expansion in rows 22-31. As before, we begin by analyzing expansion performance using shingle-based expansion (All/Nugget-Item/Shingle). Comparing against no expansion, we observe that automatic nugget-item expansion over all sentences with shingle similarity does decrease system ranking error, but only by a small margin. For instance, when the threshold=0.70, the number of rank swaps needed to restore the correct ranking only drops by 0.1052 (1.5789 to 1.4737). The resultant performance is similar to item-item shingle based expansion over all sentences discussed above, although we see that nugget-item

expansion performance is more stable with respect to the expansion threshold than when the item-item linking is used. On the other hand, when switching to semantic similarity for nugget-item expansion (All/Nugget-Item/Semantic), we observe a different pattern. In particular, we see that ranking errors decrease rapidly as we lower the threshold until we reach θ=0.96, below which ranking error increases. Indeed, item-nugget semantic expansion over the all set where θ=0.96 is the overall most effective expansion approach (tied with item-item semantic expansion applied to only relevant sentences) with an Avg Rank Swaps of 0.6842 and a $a\tau_{AP}$ of 0.9878 (a statistically significant increase in correlation over no expansion).

Summary: To answer RQ2, we have shown that automatic <sentence,nugget> expansion techniques that find textually or semantically similar sentences and use those sentences to infer matches can be effective for increasing the robustness of the TREC-TS test collections. On the other hand, of the approaches tested, we have observed significant gains only in two specific cases. First, item-item-based expansion performed well, but only when we restricted expansion to start from known relevant sentences. Second, nugget-item expansion performed well only when using semantic similarity rather than shingle-based similarity. Interestingly, these are very different expansion approaches, but appear to both improve the robustness of the test collections to a similar degree. In the following section we examine the proposed expansion approaches in more detail based on the matches (<sentences,nugget> pairs) that they add to the ground truth, to evaluate what they are expanding and what limitations that they have.

6 RQ3: WHAT MAKES A GOOD EXPANSION APPROACH?

In the previous section, we showed that using automatic expansion of <sentences,nugget> matches can increase the robustness of the TREC-TS test collections, as illustrated by the reduction in the number of mis-rankings of systems. However, while the previous experiment enables us to quantify the practical effect of automatic expansion, it is also important to examine why this occurs, such that we can understand whether there are cases where the expansion approaches shown to be effective may fail and why the poorly performing expansion approaches go wrong. Hence, in this section we examine the matches that our automatic expansion approaches are adding to the ground truth in more detail. In particular, we first describe a series of metrics that we can use to quantify the quality of the <sentences,nugget> matches that are being added via expansion in Section 6.1. Meanwhile, we report the performance of each expansion technique in terms of these metrics in Section 6.2.

6.1 Match Expansion Metrics

To examine the quality of the <sentence,nugget> pairs that are added by expansion, we first need to define effective quality metrics. As we are proposing automatic methods to expand the ground truth, we could be doing more harm than good by introducing false <sentence,nugget> pairs. Indeed, as we observed previously in Section 5.4, some of the expansion approaches that we tested harmed the system ranking performance of the test collections, indicating that these expansion approaches were adding noise to the ground truth. Recall from Section 5.1 that we are using a depooling methodology to compare scenarios where a system is included in the pool vs. when that system is excluded from the pool. By depooling a system, we in effect remove a set of matches (<sentence,nugget> pairs) from the ground truth. An automatic expansion attempt will restore a portion of these matches. Hence, we can evaluate how successful an expansion attempt is based on the proportion of these matches that are restored. In this case, we can measure how effective a restoration attempt is in two ways: 1) how many of the depooled sentences were restored; 2) how many of the full <sentence,nugget> pairs were restored. We make this distinction such that we

can quantify the impact of the Similarity Metric (that will mainly affect the number of sentences restored) and the impact of the Linking Strategy (that will mainly contribute to the pairing of sentences and nuggets). As such we define the following metrics to evaluate the quality of the <sentence,nugget> pairs produced by expansion:

Avg # Restored/Missed: When considering a <sentence,nugget> pair produced by expansion we first want to understand whether the sentence within that pair is one of those removed by depooling, i.e. one of those we aimed to restore. To capture this, we report the raw number of sentences that were removed by depooling that were subsequently restored by expansion. This is averaged over the 19 depooling scenarios. We denote this *Avg # Restored*. The larger the number of restored sentences the better. However, to understand how effective the expansion is, we also need to quantify how many we could have restored. For this reason, we also report *Avg # Missed*, i.e. the number of sentences that we failed to restore (again averaged over the 19 depooling scenarios). These two metrics are defined as follows:

$$\#\ \text{Restored}(\mathcal{S}_p^{\mathcal{M}}, \mathcal{S}_{up}^{\mathcal{M}}, \mathcal{S}_{up\rightarrow e}^{\mathcal{M}}) = |(\mathcal{S}_p^{\mathcal{M}} \cap \mathcal{S}_{up}^{\mathcal{M}}) \cap (\mathcal{S}_{up\rightarrow e}^{\mathcal{M}} \cap \mathcal{S}_{up}^{\mathcal{M}})| \tag{5}$$

$$\#\ \text{Missed}(\mathcal{S}_p^{\mathcal{M}}, \mathcal{S}_{up}^{\mathcal{M}}, \mathcal{S}_{up\rightarrow e}^{\mathcal{M}}) = |(\mathcal{S}_p^{\mathcal{M}} \cap \mathcal{S}_{up}^{\mathcal{M}})| - \#Restored(\mathcal{S}_p^{\mathcal{M}}, \mathcal{S}_{up}^{\mathcal{M}}, \mathcal{S}_{up\rightarrow e}^{\mathcal{M}}) \tag{6}$$

where $\mathcal{S}_p^{\mathcal{M}}$ is the set of sentences that correctly matched one or more nuggets if system $\mathcal{S}$ was pooled, $\mathcal{S}_{up}^{\mathcal{M}}$ is the set of sentences that correctly matched one or more nuggets even if system $\mathcal{S}$ was not pooled (i.e. derived from sentences contributed by other pooled systems) and $\mathcal{S}_{up\rightarrow e}^{\mathcal{M}}$ is the set of sentences that would correctly match one or more nuggets if system $\mathcal{S}$ was not pooled but after a matching expansion technique (see Section 5.3) has been applied.

Avg # Unknown: When we consider all sentences contained within the stream for expansion (the 'All' item-set), it is likely that we will introduce sentences into the ground truth that are neither 'restored' or 'missed', i.e. they are new sentences that have never been pooled/assessed before. For this reason, we do not know whether these sentences are part of good <sentence,nugget> pairs. We refer to these as 'unknown' sentences, and will examine them in more depth later in Section 7. However, we do report the number of these unknown sentences that are introduced by expansion for future reference. As before, this is averaged over the 19 depooling scenarios. We denote this metric as *Avg # Unknown* and is calculated as follows:

$$\#\ \text{Unknown}(\mathcal{S}_p^{\mathcal{M}}, \mathcal{S}_{up\rightarrow e}^{\mathcal{M}}) = |\mathcal{S}_{up\rightarrow e}^{\mathcal{M}} \cap \{\mathcal{S}_p^{\mathcal{M}}\}^C| \tag{7}$$

where $\{\mathcal{S}_p^{\mathcal{M}}\}^C$ is the relative complement of $\mathcal{S}_p^{\mathcal{M}}$.

Expansion Recall (*E-Recall*): While the Avg # Restored/Missed metrics together capture the proportion of sentences restored, it is valuable to have a single metric, hence we define E-Recall as the proportion of missing sentences that were correctly matched to one or more nuggets, calculated as:

$$\text{E-Recall}(\mathcal{S}_p^{\mathcal{M}}, \mathcal{S}_{up}^{\mathcal{M}}, \mathcal{S}_{up\rightarrow e}^{\mathcal{M}}) = \frac{|(\mathcal{S}_p^{\mathcal{M}} \cap \mathcal{S}_{up}^{\mathcal{M}}) \cap (\mathcal{S}_{up\rightarrow e}^{\mathcal{M}} \cap \mathcal{S}_{up}^{\mathcal{M}})|}{|\mathcal{S}_p^{\mathcal{M}} \cap \mathcal{S}_{up}^{\mathcal{M}}|} \tag{8}$$

This is analogous to recall from a classification perspective, representing the proportion of all sentences that we were able to correctly restore through automatic expansion. Note that we are

not interested in a precision-like metric here, as 'false positives' represent sentences that do not exist in $\mathcal{S}_p^{\mathcal{M}}$ (i.e. are the 'unknown' sentences which we do not have assessments for).

Avg. Expansion Pair F_1 (*aEP-F_1*): Having defined metrics that quantify how effectively an expansion approach restores the depooled sentences, we also need a metric that evaluates whether the <sentence,nugget> pairs are correct. Indeed, an expansion approach may be effective at identifying relevant sentences to expand with, but then fails to match those sentences to the correct nugget(s). Extending the idea of precision and recall for this setting, we start by defining Expansion Pair Precision (*EP-P*) and Expansion Pair Recall (*EP-R*) for a sentence u as follows:

$$\text{EP-P}(\mathcal{M}_p^u, \mathcal{M}_{up\rightarrow e}^u) = \frac{|\mathcal{M}_p^u \cap \mathcal{M}_{up\rightarrow e}^u|}{|\mathcal{M}_{up\rightarrow e}^u|} \tag{9}$$

$$\text{EP-R}(\mathcal{M}_p^u, \mathcal{M}_{up\rightarrow e}^u) = \frac{|\mathcal{M}_p^u \cap \mathcal{M}_{up\rightarrow e}^u|}{|\mathcal{M}_p^u|} \tag{10}$$

where $\mathcal{M}_p^u$ is the set of <sentence,nugget> pairs for sentence u that resulted from matching after being pooled and $\mathcal{M}_{up\rightarrow e}^u$ is the set of <sentence,nugget> pairs for sentence u that were produced by expansion when u was not pooled. *EP-P* measures the proportion of nugget matches for sentence u produced by expansion that were correct, while *EP-R* measures the proportion of all nugget matches for sentence u that were restored. We can average both *EP-P* and *EP-R* across all sentences with matches from the TREC-TS pool (i.e. where we know $\mathcal{M}_p^u$) and that were subject to expansion (i.e. $\mathcal{M}_p^u$ and $\mathcal{M}_{up\rightarrow e}^u$ are different), which we denote as *aEP-F* and *aEP-R*, respectively. In our later experiments, to have a single metric representing pair generation quality, we report the harmonic mean of *aEP-P* and *aEP-R* across sentences, denoted *aEP-F_1*:

$$\text{aEP-F}_1 = 2 \cdot \frac{\text{aEP-P} \cdot \text{aEP-R}}{\text{aEP-P} + \text{aEP-R}} \tag{11}$$

Using these metrics, we can express how effective an expansion attempt is. For instance, if we have an expansion method that has *Avg # Restored/Missed/Unknown* of 1.0222/5.0047/23.4521, *E-Recall* of 0.1666 and an *aEP-F_1* of 0.9621, then we can read this follows. First, the expansion method correctly restored around 1 sentence on average per topic/event, it failed to restore another 5 sentences per topic/event that had been removed by depooling, and it added approximately another 23 sentences per topic/event which we don't know whether they are correct or not (i.e. they are unknown sentences). The proportion of depooled sentences that were restored was 16.6%. Meanwhile, considering the full <sentence,nugget> pairs rather than just the sentences, matching F_1 was high at 0.9621, meaning that the <sentence,nugget> pairs added by expansion were nearly all correct (we predominantly did not match sentences to nuggets that they did not cover).

6.2 Evaluating Match Quality

Having defined how we can evaluate the quality of the matches introduced into the ground truth via expansion, we now evaluate the matches produced by each of the expansion approaches reported previously in Section 5.4. Table 7 reports the Avg # Restored/Missed/Unknown sentences produced by each expansion approach, as well as overall sentence restoration performance (E-Recall) and matching performance (aEP-F_1). For ease of reference, the impact column of Table 7 provides a summary of the impact that each expansion approach had on text collection robustness based on the results reported in Section 5.4, where ▲ ▲/▲/-/▼/▼ ▼ indicates strong positive/positive/neutral/negative/strong negative impact respectively. We aim to examine what makes the best approaches identified in Section 5.4 (those indicated by ▲ ▲) effective.

Table 7. Match expansion performance on average across depooling scenarios and topics when performing item-to-item and nugget-to-item similarity expansion. The impact column provides a summary of the impact that the expansion attempt had on text collection robustness based on the results reported in Section 5.4, where ▲▲/▲/-/▼/▼▼ indicates strong positive/positive/neutral/negative/strong negative impact respectively.

Expansion Technique				Match Expansion Performance					Impact
Item-Set	Linking	Similarity	Threshold θ	Avg # Restored	Avg # Missed	Avg # Unknown	E-Recall	aEP-F_1	
Relevant	Item-Item	Levenshtein	0.99	0.4187	5.6082	0	0.0714	0.9401	-
Relevant	Item-Item	Levenshtein	0.90	0.6094	5.4175	0	0.1096	0.8336	▲
Relevant	Item-Item	Levenshtein	0.80	0.6830	5.3439	0	0.1199	0.8349	▲
Relevant	Item-Item	Levenshtein	0.70	0.6830	5.3439	0	0.1199	0.8349	▲
Relevant	Item-Item	Semantic	1.00	0.4070	5.6199	0	0.0679	0.9662	▲
Relevant	Item-Item	Semantic	0.98	0.7591	5.2678	0	0.1239	0.8653	▲▲
Relevant	Item-Item	Semantic	0.96	1.0222	5.0047	0	0.1666	0.7436	▲▲
Relevant	Item-Item	Semantic	0.94	1.5380	4.4889	0	0.2383	0.4819	▼
Relevant	Item-Item	Semantic	0.92	2.3942	3.6327	0	0.3678	0.2253	▼▼
Relevant	Item-Item	Semantic	0.90	3.3427	2.6842	0	0.5034	0.1341	▼▼
All	Item-Item	Shingle	1.00	0.4994	5.5275	46.7556	0.0800	0.8108	-
All	Item-Item	Shingle	0.90	0.6222	5.4047	68.4140	0.1047	0.7439	▼
All	Item-Item	Shingle	0.80	0.7029	5.3240	107.0807	0.1162	0.7110	-
All	Item-Item	Shingle	0.70	0.8246	5.2023	135.6655	0.1344	0.6292	-
All	Item-Item	Shingle	0.60	0.9205	5.1064	171.3825	0.1511	0.5919	-
All	Item-Item	Semantic	1.00	0.4070	5.6199	26.5965	0.0679	0.9154	▲
All	Item-Item	Semantic	0.98	0.7591	5.2678	58.8246	0.1240	0.8198	▼
All	Item-Item	Semantic	0.96	1.0222	5.0047	94.3778	0.1666	0.7436	-
All	Item-Item	Semantic	0.94	1.5380	4.4889	168.3029	0.2384	0.4819	▼▼
All	Nugget-Item	Shingle	1.00	0.0281	5.9988	5.9778	0.0036	0.5711	-
All	Nugget-Item	Shingle	0.90	0.0573	5.9696	11.0444	0.0066	0.5478	-
All	Nugget-Item	Shingle	0.80	0.1029	5.9240	25.0667	0.0131	0.4871	-
All	Nugget-Item	Shingle	0.70	0.1263	5.9006	32.7778	0.0155	0.4628	-
All	Nugget-Item	Shingle	0.60	0.1556	5.8713	42.9333	0.0189	0.5666	-
All	Nugget-Item	Shingle	0.50	0.2070	5.8199	63.7778	0.0243	0.5238	-
All	Nugget-Item	Semantic	1.00	0.0035	6.0234	1.0889	0.0006	0.0000	-
All	Nugget-Item	Semantic	0.98	0.7591	5.2678	0.3170	0.1240	0.8198	▲▲
All	Nugget-Item	Semantic	0.96	1.0222	5.0047	0.3754	0.1666	0.7436	▲▲
All	Nugget-Item	Semantic	0.94	1.5380	4.4889	0.4327	0.2384	0.4819	▼

From Table 7, examining the number of sentences restored, we observe that across all of the expansion approaches, relatively few of the sentences removed by depooling are being restored. For instance, for the effective All/Nugget-Item/Semantic approach with $\theta = 0.98$, only 0.7591 sentences per topic were correctly restored on average, or only 12% of those that could have been restored (E-Recall=0.1240). This behaviour is consistent across all of the expansion approaches that exhibited positive impact on test collection robustness (those indicated by ▲). This is interesting, as it indicates that we can markedly increase test collection robustness with only a small number of additions. To explain why this is the case, we need to consider the metrics under which we are ranking systems. The TREC-TS metrics described previously in Section 3.2 are primarily evaluating the proportion of nuggets for an event that a summary covers. A summary is only awarded gain once per nugget it covers (to avoid promoting redundancy in the summaries). As such, if we consider a summary S that contains three sentences A, B and C that cover nugget X, then to correctly determine whether S covers X at evaluation time, only a single match is required (either <A,X>, <B,X> or <C,X>), not all three. Considering this from the perspective of automatic expansion, it is logical therefore then that a small number of additions to the ground truth can have a large impact, since restoring a single <sentence,nugget> pair is sufficient to indicate that a summary as a whole covers a nugget.

Next, we consider the quality of the matches themselves, i.e. when we identify a relevant sentence and then link it to a nugget do we select the correct nugget(s)? This is captured by aEP-F_1. As we can see from Table 7, when we consider the expansion approaches that were shown to be effective

(those indicated by ▲), aEP-F_1 is high (ranging between 0.9401 and 0.7436). This shows that as we would expect, the ability to identify the correct nugget(s) during matching is critical. However, it is worth noting that perfect matching performance is not required, as the expansion approaches that had the highest positive impact on test collection robustness exhibit aEP-F_1 scores of around 0.75, i.e. only 25% of the added matches were the ones removed by depooling. There are two possible explanations for this. Either those 25% of matches are actually not noise, but matches that the original assessors missed when creating the ground truth originally, or the TREC-TS metrics are sufficiently robust to not be negatively impacted by the introduction of a small amount of noise into the ground truth.

Finally, it is worth considering the number of 'unknown' sentences introduced by each expansion approach, i.e. sentences that have never been pooled/assessed but the expansion approaches consider to be relevant. From Table 7, we see that the expansion approaches that start from the Relevant item-set do not introduce unknown sentences. This is expected, since to be considered a relevant sentence, it must have already been assessed. However, considering the other expansion approaches that consider all sentences as candidates for matching (All item-set), we see that these approaches can add a large number of additional sentences into the ground truth. For example, the All/Item-Item/Shingle expansion approach with θ=1.0 introduced approximately 46 new sentences per topic - dwarfing the number of restored sentences. On the other hand, considering only the highly effective expansion approaches (those indicated by ▲ ▲) we see that these approaches do not add many unknown sentences in comparison to the other expansion approaches. For instance, All/Nugget-Item/Semantic expansion with θ=0.96 only introduces 0.3170 unknown matches per topic on average. The natural conclusion to make from this would be that adding unknown sentences contributes to reduced test collection robustness (if we assume that the added unknown sentences are irrelevant). However, this does not appear to be the case, as we can see from Table 7 that some expansion approaches are introducing a large number of unknown sentences without negatively impacting the robustness of the test collection. For example All/Item-Item/Semantic expansion with θ=1.0 introduced over 26 unknown sentences on average while having a positive impact on test collection robustness. Hence, we can conclude that at least some of these unknown sentences are in fact resulting in the introduction of good quality matches. Moreover, expansion approaches such as All/Item-Item/Shingle where θ=0.6 show that extreme levels of expansion (171 sentences added per topic on average) are possible without degrading the test collection robustness. Given the large number of unknown matches being added by some of these expansion approaches, it is clear that we need to investigate these unknown matches in more detail, which we focus on in the next section.

Summary: In this section we examined why the automatic expansion approaches tested in Section 5.4 are effective. To answer RQ3, we conclude that effective expansion techniques need to have high matching accuracy when linking sentences to nuggets, as we observed rapid decreases in test collection robustness as aEP-F_1 similarly decreases. On the other hand, sentence restoration recall was shown to not be an important factor in improving robustness, as some of the best expansion approaches only restored around 15% of the missing sentences - showing that small changes to the test collection can have a large impact on robustness. Finally, we observed that while the best expansion approaches do not introduce many unknown sentences into the ground truth, it is possible to dramatically expand the ground truth set without negatively impacting test collection robustness, which we examine in more detail in the next section.

7 RQ4: SHOULD WE ALSO INTRODUCE UNKNOWN SENTENCES INTO THE GROUND TRUTH?

In the previous section we showed that while the best expansion approaches did not introduce many unknown sentences into the ground truth, it was possible to do so without harming test collection robustness. As such, a natural question to ask is, should we do so, and what advantage might this provide? To answer this question, we need to quantify whether these unknown sentences being added by the different approaches are leading to good quality matches. If these are resulting in predominately good quality matches then they will enhance the test collection, otherwise we are simply adding noise that may cause issues in the future.

From a practical perspective, to evaluate the quality of the unknown matches (<sentence,nugget> pairs), we need to assess whether those matches are correct or not. A correct match is one where the sentence contains the information represented by the nugget. By definition, unknown matches have not previously been assessed. Hence, we need to manually assess them. As such, we define a crowdsourced matching experiment, where we have human assessors judge the correctness of a sample of these unknown matches. We describe the experimental setup of the crowdsourced matching experiment in Section 7.1, while we report on the quality of the unknown matches based on the crowdsourced study in Section 7.2.

7.1 Crowdsourced Labeling of Unknown Matches

Data Sampling: From Table 7 it is clear that manually assessing all of the unknown matches is not practical due to the large volume of additional matches added by some expansion techniques. For example, considering All/Item-Item/Shingle expansion with a threshold of 0.9 would require 3,147 matches to be assessed (68.414 matches * 46 events). Hence, we need to select a sample to be assessed. For this reason, we select at random 300 unknown matches from each expansion approach. However, as we observed previously, not all expansion approaches generate unknown matches. In particular, both the approaches that use the Relevant item-set do not generate unknown matches, so we omit them from this experiment. Meanwhile, All/Nugget-Item/Semantic expansion discussed above returns almost no unknown expansions for the thresholds θ tested. Rather than omitting that expansion approach as well, we instead select deeper thresholds to test with (θ=[0.9,0.8,0.7]). This leaves us with 21 expansion approaches. The 300 matches selected are disjoint, hence this results in 6300 unknown matches to be assessed.

Labeling Task: To assess these 6300 unknown matches, we use the medium of crowdsourcing. In particular, we have crowdsourced workers manually annotate the tweets, using Amazon's Mechanical Turk.[6] Following earlier work in crowdsourced labelling [22] that indicated labelling accuracy does not significantly increase beyond 2-3 workers, each tweet-target pair is given to two distinct workers. If those workers agree, then the match is kept, the match is discarded otherwise. Each worker is shown the sentence text of the match, along with the text representation of the nugget and asked whether the information represented by the nugget is contained within the sentence (binary classification). An example of the Amazon Mechanical Turk job (known as a HIT) is shown in Figure 1, while the worker instructions are shown in Figure 2.

Crowdsourcing Configuration: Following best practices in crowdsourcing [31], we apply a series of quality assurance techniques to avoid poor-quality work. First, as discussed above, we have two assessors judge each match, resulting in a total of 12,600 HITs. Second, we only use Amazon Mechanical Turk 'Master' assessors, who have a strong track record for categorization tasks, as

[6]https://www.mturk.com/

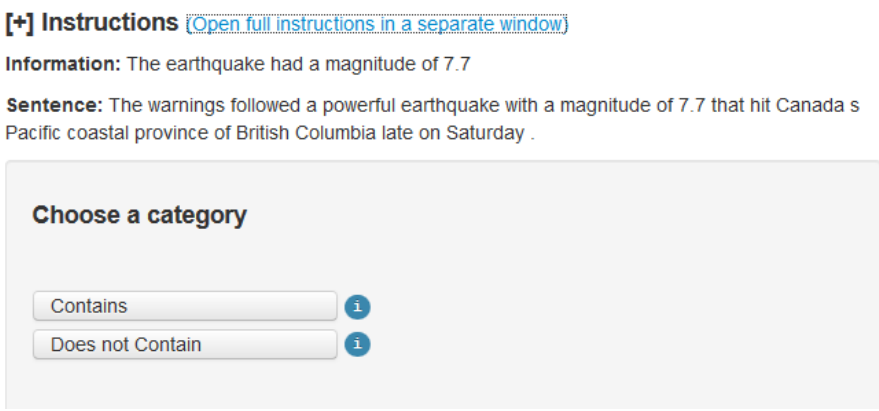

Fig. 1. An example of the Amazon Mechanical Turk Job (HIT).

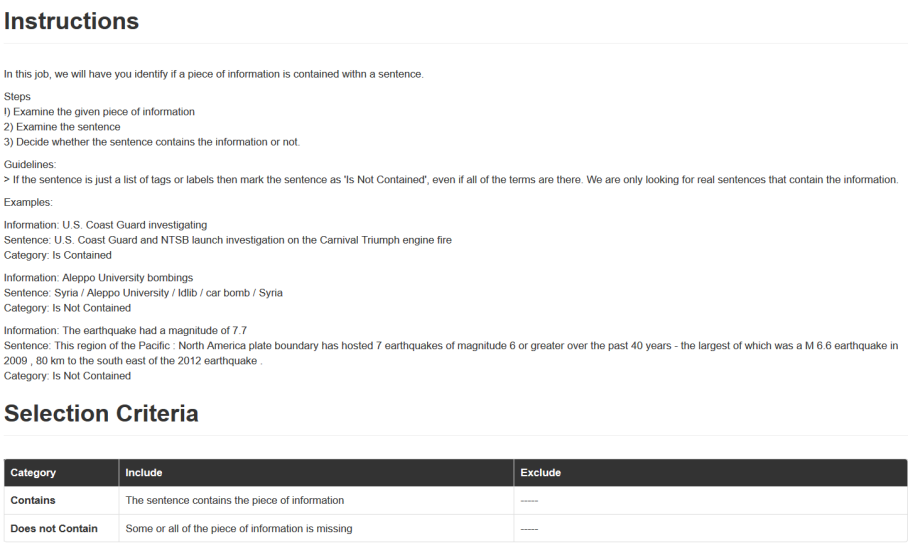

Fig. 2. Instructions given to the crowdsourced workers.

these are closer to the 'expert' NIST assessors that performed the original matching during the TREC-TS track. To avoid over biasing the dataset towards a small number of workers, we limit the number of individual HITs an individual worker can complete to 1000. We pay each assessor $0.02 U.S. per match assessed, plus Amazon's 20% fee.

Crowdsourcing Statistics: In total 162 unique workers participated in the crowdsourced matching tasks. The distribution of workers in terms of the number of HITs completed is shown in Figure 3. The agreement of the workers on the tackled HITs overall was 62%. This indicates that a significant number of the unknown matches were difficult for the assessors to label (i.e. conclusively decide whether the sentence contains the information represented by the nugget). This is expected, as the sentences can be ambiguous or in-exact. For instance, consider the following sentence:

"Earthquake Canada initially measured it at magnitude 7.1 but altered their report later Saturday."

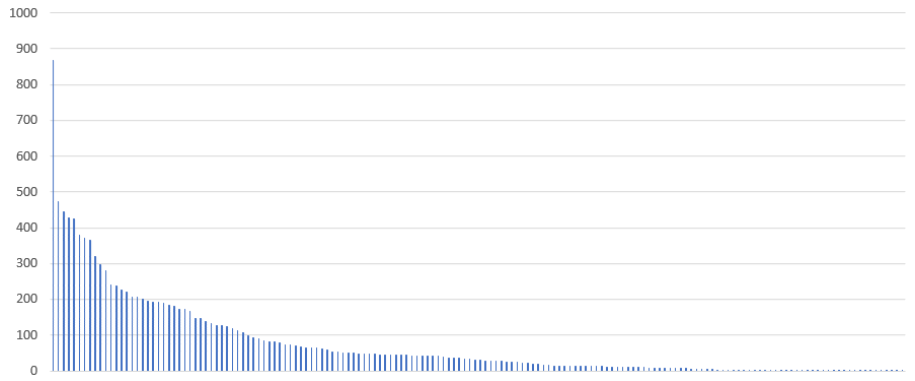

Fig. 3. Distribution of HITS completed by workers.

when matched to the nugget 'the earthquake had a magnitude of 7.7'. As we can see, the exact magnitude number is not the same, so is this a valid match? For this reason, we discard matches where the crowd workers disagreed, such as in the above case. After discarding unknown matches where workers did not agree, we are left with 3,835 unknown matches, which we use for evaluation of unknown match accuracy below.

Metrics: As unknown matches by definition were not matches pooled by our synthetic systems, we cannot evaluate their quality in terms of their impact on the system ranking. Instead, for each expansion approach, we report: the number of assessed matches produced by that expansion approach that were both judged by crowd assessors and the crowd assessors agreed (denoted *AVG # Assessed*); of those the number that were correct matches (both workers indicated that the sentence contained the information in the nugget, denoted *AVG # Correct*); the number that were incorrect matches (both workers indicated that the sentence did not contain the information in the nugget, denoted *AVG # Incorrect*); and the percentage that were correct (denoted *% Correct (AVG)*). The larger the proportion of correct matches the better the expansion technique is. Note that match expansion occurs after depooling of a system, so in effect we have 19 expansion scenarios (one per system) for each expansion approach. As with the previous experiments, we average over these 19 expansion scenarios.

7.2 Unknown Match Evaluation

Having produced crowdsourced labels for a random sample of unknown matches per expansion approach, we now examine to what extent those unknown matches were correct. We aim to evaluate whether the introduction of unknown matches into the ground truth by these expansion approaches is desirable or not. If the majority of the unknown matches are in fact good quality matches, then these are valuable additions to the test collection. But if they are predominantly incorrect (noisy) matches, then adding them may negatively impact the test collection - indeed this may further explain why some of these approaches harm test collection robustness.

Table 8 presents the results of manual assessment of the unknown matches across depooling scenarios by the crowdsourced assessors. As with the previous section, the impact column provides a summary of the impact that the expansion attempt had on text collection robustness based on the results reported in Section 5.4, where ▲ ▲/▲/-/▼/▼ ▼ indicates strong positive/positive/neutral/ negative/strong negative impact respectively. Note that in two cases we experiment with deeper

Table 8. Results when having crowdsourced assessors label 'unknown' matches produced via expansion. Results are averaged across depooling scenarios. The impact column provides a summary of the impact that the expansion attempt had on text collection robustness based on the results reported in Section 5.4, where ▲ ▲/▲/-/▼/▼ ▼ indicates strong positive/positive/neutral/negative/strong negative impact respectively. * denotes a setting that we did not previously report performance for in Section 5.4.

Expansion Approach				Sampling		Crowdsourced Assessment of Unknown Matches			Impact
Linking	Similarity	Item-Set	Threshold	# Sampled	AVG # Assessed	AVG # Correct	AVG # Incorrect	% Correct (AVG)	
Item-Item	Shingle	All	1.00	300	381.16	301.58	79.58	79%	-
Item-Item	Shingle	All	0.90	300	479.16	378.58	100.58	79%	▼
Item-Item	Shingle	All	0.80	300	648.68	515.11	133.58	79%	-
Item-Item	Shingle	All	0.70	300	759.84	601.26	158.58	79%	-
Item-Item	Shingle	All	0.60	300	868.89	675.26	193.63	78%	-
Item-Item	Shingle	All	0.50	300	961.95	749.37	212.58	78%	-
Item-Item	Semantic	All	1.00	300	305.58	237.16	68.42	78%	▲
Item-Item	Semantic	All	0.98	300	545.95	406.63	139.32	74%	▼
Item-Item	Semantic	All	0.96	300	758.32	550.26	208.05	73%	-
Item-Item	Semantic	All	0.94	300	1036.37	735.21	301.16	71%	▼ ▼
Item-Item	Semantic	All	0.92	300	1246.37	880.58	365.79	71%	▼ ▼*
Item-Item	Semantic	All	0.90	300	1446.26	1007.11	439.16	70%	▼ ▼*
Nugget-Item	Shingle	All	1.00	270	217.00	211.00	6.00	97%	-
Nugget-Item	Shingle	All	0.90	300	315.00	307.00	8.00	97%	-
Nugget-Item	Shingle	All	0.80	300	435.00	424.00	11.00	97%	-
Nugget-Item	Shingle	All	0.70	300	499.00	484.00	15.00	97%	-
Nugget-Item	Shingle	All	0.60	300	560.00	540.00	20.00	96%	-
Nugget-Item	Shingle	All	0.50	300	627.00	598.00	29.00	95%	-
Nugget-Item	Semantic	All	0.90	300	174.00	77.00	97.00	44%	▼ ▼*
Nugget-Item	Semantic	All	0.80	300	601.00	349.00	252.00	58%	▼ ▼*
Nugget-Item	Semantic	All	0.70	300	1077.00	627.00	450.00	58%	▼ ▼*

thresholds than were reported in Section 5.4, for consistency we provide impact indicators for these approaches as well, and highlight them with *.

From Table 8 we observe the following points of interest. First, we see that the Shingle similarity-based expansion approaches are very effective at finding new correct matches. Indeed, the best performing expansion approach in terms of the unknown matches is All/Nugget-Item/Shingle expansion, where over 95% of the generated unknown matches were deemed correct by our crowd workers. Meanwhile, around 80% of the unknown matches produced by All/Item-Item/Shingle expansion were judged correct. However, despite adding all of these new correct matches to the ground truth, their impact on test collection robustness appears to be minimal. Relating this to the previous discussion on TREC-TS metrics, we can explain this in terms of redundant matching. Lets assume that we have a summary S that contains a sentences A, B and C that all cover the information in a nugget X. Lets further assume that in the ground truth (pre-expansion) we already have a match between the sentence A and nugget X, <A,X>. This is sufficient information to evaluate summary S given X. So even if we perform automatic expansion and manage to correctly add <B,X> and <C,X>, then this will not impact the scoring of S. None the less, this can be seen as an enhancement to the test collection, as in cases where a summary did not contain sentence A but did contain sentence B or C, the additional <B,X> and <C,X> matches would enable a more accurate performance estimation of that summary.

On the other hand, examining the expansion approaches that build on semantic similarity, we see that the proportion of unknown matches judged correct is lower than that seen when using shingle-based similarity. Indeed, we observe that All/Item-Item/Semantic performance quickly drops below 75% correct as the selection threshold is relaxed. Meanwhile, given the deeper thresholds we tested for All/Nugget-Item/Semantic expansion, we see that % Correct is below 60%. This provides a further explanation for why these expansion approaches can lead to reductions in test collection robustness, as they are adding a large number of incorrect matches into the ground truth.

Summary: From the results of our crowdsourced evaluation of unknown matches, we showed that the unknown matches produced by approaches leveraging shingle-based similarity are predominantly correct, e.g. around 95% correct for All/Nugget-Item/Shingle expansion. Hence, to answer RQ4, we conclude that it would be advantageous to introduce these unknown matches into the ground truth, as while they don't appear to have an immediate impact on test collection robustness based on the synthetic systems used for testing here, they may enable better performance estimations for other timeline generation systems.

8 CONCLUSIONS

In this paper, we quantified the extent to which the TREC Temporal Summarization (TREC-TS) test collections are able to robustly rank different timeline generation systems in scenarios where one of those systems was not included in the sampled pool. We also proposed a new framework for the automatic expansion of those test collections aimed at improving their robustness, analysed different expansion techniques, and evaluated their impact on the robustness of TREC-TS 2013, 2014 and 2015 test collections.

Through a leave-one-out (depooling) experiment, we observed a mixed picture in terms of test collection robustness. In particular, the TREC-TS test collections appear to be robust when evaluating systems that were not pooled at the lower end of the effectiveness scale, i.e. the correlation with the correct system ranking is perfect when the system is depooled. On the other hand, when more effective systems were depooled, we started to observe ranking errors and lower correlations, particularly towards the top ranks. Hence, we conclude that using the TREC-TS test collections out-of-the-box is subject to some risk.

To reduce this risk, we proposed a three component framework (comprising a Linking Strategy, Similarity Metric and Item Set) that uses the available sentence pool and associated matches (<sentence,nugget> pairs) to automatically generate new matches that might have been missed due to those sentences not being pooled. In particular, we experimented with two linking strategies (item-item vs. nugget-item), three similarity metrics (levenshtein distance, shingle similarity and semantic similarity) and two item sets (relevant vs. all). Our experiments using a range of instantiations of this framework show that automatically adding even a small number of <sentence,nugget> pairs can markedly reduce the number of ranking errors observed when using the test collections. In particular, we found that the best framework configurations can reduce ranking errors by between 30% to 50% and improve correlation against the correct ranking of systems by a statistically significant margin. Furthermore, by analyzing the matches produced by different framework instantiations, we showed that (as we might expect) to improve test collection robustness, accuracy when linking items to nuggets is critical, but that surprisingly only a small number of added matches are needed to enhance test collection robustness significantly (the best instantiations tested only exhibited around 15% match expansion recall). Finally, through a crowdsourced study examining previously unassessed matches produced by the framework, we observed that instantiations that used Shingle-based similarity were very effective at generating large volumes of correct matchings that were not in the ground truth.

Overall, our work highlights the limitations of shallow pooling for creating test collections from large streaming data, particularly for tasks where item selections are not independent (in this case selection of an item for inclusion within the summary is dependent on past selections of other sentences), resulting in low overlap between systems when pooling summaries for labeling. Our results show that automatic matching expansion techniques can help mitigate this issue and enhance the robustness of the TREC-TS test collections. As such, we recommend the use of expanded match sets when evaluating new systems, particularly in cases where low completeness levels are observed. Furthermore, we see additional use-cases for these automatic expansion techniques

in other domains that exhibit similar characteristics to TREC-TS, i.e. cases where high-volume datasets/data streams are being used that contain significant redundancy in content but where limited assessment resources are available. For instance, datasets created for social media processing tasks like Twitter search are prime candidates for the proposed expansion techniques discussed here.

9 DATA RELEASE

To support future researchers working with the TREC-TS test collections, we provide both the synthetic systems that we used in this study as potential new baselines for the community, as well as the expanded assessment matches produced by the best performing expansion approach (Relevant/Item-Item/Semantic Expansion $\rightarrow \theta$=0.96), which we recommend that researchers use in scenarios where they are experiencing low completeness levels under the official labels. These can be freely downloaded at:

http://dx.doi.org/10.5525/gla.researchdata.613

We also release the expanded set of matches produced by the All/Nugget-Item/Shingle expansion approach, as our crowdsourced study showed that these are also highly accurate matches. Even though they did not appear to markedly enhance test collection robustness, they may be useful to future researchers. These can be downloaded at:

http://dx.doi.org/10.5525/gla.researchdata.743

10 ACKNOWLEDGEMENTS

This work was funded as part of the Incident Streams Project within the Public Safety Communications Research Program, by the National Institute of Standards and Technology (U.S.).

REFERENCES

[1] James Allan, Rahul Gupta, and Vikas Khandelwal. 2001. Temporal summaries of new topics. In *ACM SIGIR 2001*.
[2] Javed Aslam, Fernando Diaz, Matthew Ekstrand-Abueg, Richard McCreadie, Virgil Pavlu, and Tetsuya Sakai. 2014. TREC 2014 Temporal Summarization Track Guidelines. In *TREC 2014*.
[3] Javed Aslam, Fernando Diaz, Matthew Ekstrand-Abueg, Richard McCreadie, Virgil Pavlu, and Tetsuya Sakai. 2015. TREC 2015 Temporal Summarization Track Overview. In *TREC 2015*.
[4] Javed A Aslam, Matthew Ekstrand-Abueg, Virgil Pavlu, Fernando Diaz, and Tetsuya Sakai. 2013. TREC 2013 Temporal Summarization. In *TREC 2013*.
[5] Gaurav Baruah, Richard McCreadie, and Jimmy Lin. 2017. A Comparison of Nuggets and Clusters for Evaluating Timeline Summaries. In *ACM CIKM 2017*.
[6] Gaurav Baruah, Adam Roegiest, and Mark D. Smucker. 2014. The Effect of Expanding Relevance Judgements with Duplicates. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '14)*. ACM, New York, NY, USA, 1159–1162. https://doi.org/10.1145/2600428.2609534
[7] Eric Breck, John D. Burger, Lisa Ferro, Lynette Hirschman, David House, Marc Light, and Inderjeet Mani. 2000. How to Evaluate your Question Answering System Every Day and Still Get Real Work Done. *CoRR* cs.CL/0004008 (2000). http://arxiv.org/abs/cs.CL/0004008
[8] Andrei Z. Broder. 2000. Identifying and Filtering Near-Duplicate Documents. In *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching (COM '00)*. Springer-Verlag, Berlin, Heidelberg, 1–10. http://dl.acm.org/citation.cfm?id=647819.736184
[9] Chris Buckley and Ellen M. Voorhees. 2004. Retrieval Evaluation with Incomplete Information. In *ACM SIGIR 2004*.
[10] Chris Buckley and Ellen M. Voorhees. 2017. Evaluating Evaluation Measure Stability. *SIGIR Forum* 51, 2 (Aug. 2017), 235–242.
[11] Ben Carterette and James Allan. 2007. Semiautomatic Evaluation of Retrieval Systems Using Document Similarities. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management (CIKM '07)*. ACM, New York, NY, USA, 873–876. https://doi.org/10.1145/1321440.1321564

[12] John M. Conroy, Judith D. Schlesinger, and Dianne P. O'Leary. 2011. Nouveau-ROUGE: A Novelty Metric for Update Summarization. *Computational Linguistics* 37 (2011), 1–8.
[13] Hoa Dang. 2005. Overview of DUC 2005. In *DUC 2015*.
[14] Hoa Trang Dang and Karolina Owczarzak. 2008. Overview of the TAC 2008 Opinion Question Answering and Summarization Tasks. In *TAC 2008*.
[15] Matthew Ekstrand-Abueg, Richard McCreadie, Virgil Pavlu, and Fernando Diaz. 2016. A Study of Realtime Summarization Metrics. In *CIKM 2016*.
[16] Matthew Paul Ekstrand-Abueg. 2016. A Comprehensive Method for Automating Test Collection Creation and Evaluation for Retrieval and Summarization Systems (Phd Thesis). In *Computer Science Dissertations*. Northeastern University, Boston, MA, USA. https://repository.library.northeastern.edu/files/neu:cj82pp41d
[17] Qi Guo, Fernando Diaz, and Elad Yom-Tov. 2013. Updating Users about Time Critical Events. In *ECIR 2013*.
[18] Daniel S. Hirschberg. 1975. A linear space algorithm for computing maximal common subsequences. *Commun. ACM* 18, 6 (1975), 341–343.
[19] Chris Kedzie, Fernando Diaz, and Kathleen McKeown. 2016. Real-Time Web Scale Event Summarization Using Sequential Decision Making. *arXiv preprint arXiv:1605.03664* (2016).
[20] Chris Kedzie, Kathleen McKeown, and Fernando Diaz. 2015. Predicting Salient Updates for Disaster Summarization.. In *ACL 2015*.
[21] Tom Kenter and Maarten De Rijke. 2015. Short text similarity with word embeddings. In *ACM CIKM 2015*.
[22] Svetlana Kiritchenko and Saif M Mohammad. 2016. Capturing Reliable Fine-Grained Sentiment Associations by Crowdsourcing and Best-Worst Scaling.. In *American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL)*. 811–817.
[23] Steve Krenzel. 2010 (last accessed 29-Jun-2018). *Finding Blurbs*. http://www.stevekrenzel.com/articles/blurbs
[24] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *ACL Workshop On Text Summarization 2004*.
[25] Jimmy Lin and Dina Demner-Fushman. 2005. Automatically Evaluating Answers to Definition Questions. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT '05)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 931–938. https://doi.org/10.3115/1220575.1220692
[26] Jimmy Lin and Dina Demner-Fushman. 2006. Will Pyramids Built of Nuggets Topple over?. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL '06)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 383–390. https://doi.org/10.3115/1220835.1220884
[27] Jimmy Lin, Miles Efron, Yulu Wang, and Garrick Sherman. 2014. Overview of the TREC-2014 Microblog Track. In *TREC 2014*.
[28] Jimmy J. Lin, Eileen G. Abels, Dina Demner-Fushman, Douglas W. Oard, Philip Wu, and Yejun Wu. 2005. A Menagerie of Tracks at Maryland: HARD, Enterprise, QA, and Genomics, Oh My!. In *Proceedings of the Fourteenth Text REtrieval Conference, TREC 2005, Gaithersburg, Maryland, USA, November 15-18, 2005*, Ellen M. Voorhees and Lori P. Buckland (Eds.), Vol. Special Publication 500-266. National Institute of Standards and Technology (NIST). http://trec.nist.gov/pubs/trec14/papers/umaryland-lin.hard.ent.qa.geo.pdf
[29] Craig Macdonald and Iadh Ounis. 2011. The influence of the document ranking in expert search. *Information Processing & Management* 47, 3 (2011), 376–390.
[30] Gregory Marton and Alexey Radul. 2006. Nuggeteer: Automatic Nugget-based Evaluation Using Descriptions and Judgements. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL '06)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 375–382. https://doi.org/10.3115/1220835.1220883
[31] Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2013. Identifying top news using crowdsourcing. *Information Retrieval* 16, 2 (2013), 179–209.
[32] Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2014. Incremental Update Summarization: Adaptive Sentence Selection based on Prevalence and Novelty. In *CIKM 2014*.
[33] Richard McCreadie, Rodrygo Santos, Craig Macdonald, and Iadh Ounis. 2017. Explicit diversification of event aspects for temporal summarization. *ACM TOIS* (2017).
[34] Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Sujian Li, and Houfeng Wang. 2012. Entity-centric topic-oriented opinion summarization in twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 379–387.
[35] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[36] Ani Nenkova and Kathleen McKeown. 2011. Automatic Summarization. *FnTIR* 5, 2–3 (2011), 103–233.
[37] Paul Over. 1997. TREC-6 Interactive Report. In *TREC 1997*.

[38] Virgil Pavlu, Shahzad Rajput, Peter B. Golbus, and Javed A. Aslam. 2012. IR System Evaluation Using Nugget-based Test Collections. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM '12)*. ACM, New York, NY, USA, 393–402. https://doi.org/10.1145/2124295.2124343

[39] Shahzad Rajput. 2012. A Nugget-based Test Collection Construction Paradigm (Phd Thesis). In *Computer Science Dissertations*. Northeastern University, Boston, MA, USA. https://repository.library.northeastern.edu/files/neu:904

[40] Shahzad Rajput, Matthew Ekstrand-Abueg, Virgil Pavlu, and Javed A. Aslam. 2012. Constructing Test Collections by Inferring Document Relevance via Extracted Relevant Information. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM '12)*. ACM, New York, NY, USA, 145–154. https://doi.org/10.1145/2396761.2396783

[41] Shahzad Rajput, Virgil Pavlu, Peter B. Golbus, and Javed A. Aslam. 2011. A Nugget-based Test Collection Construction Paradigm. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM '11)*. ACM, New York, NY, USA, 1945–1948. https://doi.org/10.1145/2063576.2063861

[42] Tetsuya Sakai. 2008. Comparing metrics across TREC and NTCIR:: the robustness to pool depth bias. In *ACM SIGIR 2008*.

[43] Rodrygo LT Santos, Iadh Ounis, and Craig Macdonald. 2015. Search result diversification. *Foundations and Trends in Information Retrieval* 9, 1 (2015), 1–90.

[44] Pranab Kumar Sen. 1968. Estimates of the regression coefficient based on Kendall's tau. *Journal of the American statistical association* 63, 324 (1968), 1379–1389.

[45] Luchen Tan, Adam Roegiest, Charles L. A. Clarke, and Jimmy Lin. 2016. Simple Dynamic Emission Strategies for Microblog Filtering. In *ACM SIGIR 2016*.

[46] Andrew Turpin and Falk Scholer. 2006. User performance versus precision measures for simple search tasks. In *ACM SIGIR 2006*.

[47] Ellen M. Voorhees. 1998. Variations in Relevance Judgments and the Measurement of Retrieval Effectiveness. In *SIGIR 1998*.

[48] Ellen M. Voorhees. 2003. Overview of the TREC 2003 Question Answering Track. In *TREC 2003*.

[49] Ellen M. Voorhees. 2004. Overview of the TREC 2004 Question Answering Track. In *TREC 2004*.

[50] Ellen M. Voorhees. 2005. Overview of the TREC 2005 Question Answering Track. In *TREC 2005*.

[51] Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. 2011. Evolutionary Timeline Summarization: a Balanced Optimization Framework via Iterative Substitution. In *ACM SIGIR 2011*.

[52] Emine Yilmaz, Javed A. Aslam, and Stephen Robertson. 2008. A New Rank Correlation Coefficient for Information Retrieval. In *ACM SIGIR 2008*.

[53] Emine Yilmaz, Javed A Aslam, and Stephen Robertson. 2008. A new rank correlation coefficient for information retrieval. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 587–594.

[54] ChengXiang Zhai, William W. Cohen, and John Lafferty. 2003. Beyond Independent Relevance: Methods and Evaluation Metrics for Subtopic Retrieval. In *ACM SIGIR 2003*.

[55] Chunyun Zhang, Zhanyu Ma, Jiayue Zhang, Weiran Xu, and Jun Guo. 2015. A Multi-level System for Sequential Update Summarization. In *QSHINE 2015*.

[56] Justin Zobel. 1998. How Reliable Are the Results of Large-scale Information Retrieval Experiments?. In *ACM SIGIR 1998*.